



PUBLIC
2024-04-18

SAP BTP Connectivity

Content

- 1 Connectivity 4**
- 1.1 Connectivity in the Cloud Foundry Environment. 8
 - What Is SAP BTP Connectivity?. 9
 - What's New for Connectivity. 16
 - Administration 58
 - Developing Applications. 214
 - Security. 335
 - Monitoring and Troubleshooting. 335
 - Resilience Recommendations. 337
 - On-Premise Connectivity in the Kyma Environment. 339
- 1.2 Cloud Connector. 343
 - Installation. 349
 - Configuration. 387
 - Administration. 628
 - Security. 703
 - Upgrade 715
 - Update the Java VM 716
 - Uninstallation. 718
 - Frequently Asked Questions 719
 - REST APIs. 728
- 1.3 Connectivity Proxy for Kubernetes. 732
 - Concepts. 733
 - Lifecycle Management. 764
 - Verification and Testing. 785
 - Monitoring. 787
 - Using the Connectivity Proxy. 788
 - Troubleshooting. 792
 - Frequently Asked Questions. 799
- 1.4 Transparent Proxy for Kubernetes. 800
 - Using the Transparent Proxy. 802
 - Multitenancy. 815
 - Destination Custom Resource Scope. 816
 - Integration with SAP BTP Connectivity. 817
 - Lifecycle Management. 819
 - Monitoring. 840
 - Troubleshooting. 841

	Recommended Actions.	854
	Resilience.	857
	Verification and Testing.	858
	Transparent Proxy in the Kyma Environment.	860
	Dynamic Lookup of Destinations.	866
1.5	Connectivity via Reverse Proxy.	867
1.6	Connectivity Support.	869
	Release and Maintenance Strategy.	871

1 Connectivity

SAP BTP Connectivity: overview, features, restrictions.

ⓘ Note

This documentation refers to SAP BTP, Cloud Foundry environment. If you are looking for information about the Neo environment, see [Connectivity for the Neo Environment](#).

Content

In this Topic

Hover over the elements for a description. Click an element for more information.

Overview

Features

Restrictions

- [Overview \[page 5\]](#)
- [Features \[page 6\]](#)
- [Restrictions \[page 6\]](#)

In this Guide

Hover over the elements for a description. Click an element for more information.

Cloud Foundry Environment

Cloud Connector

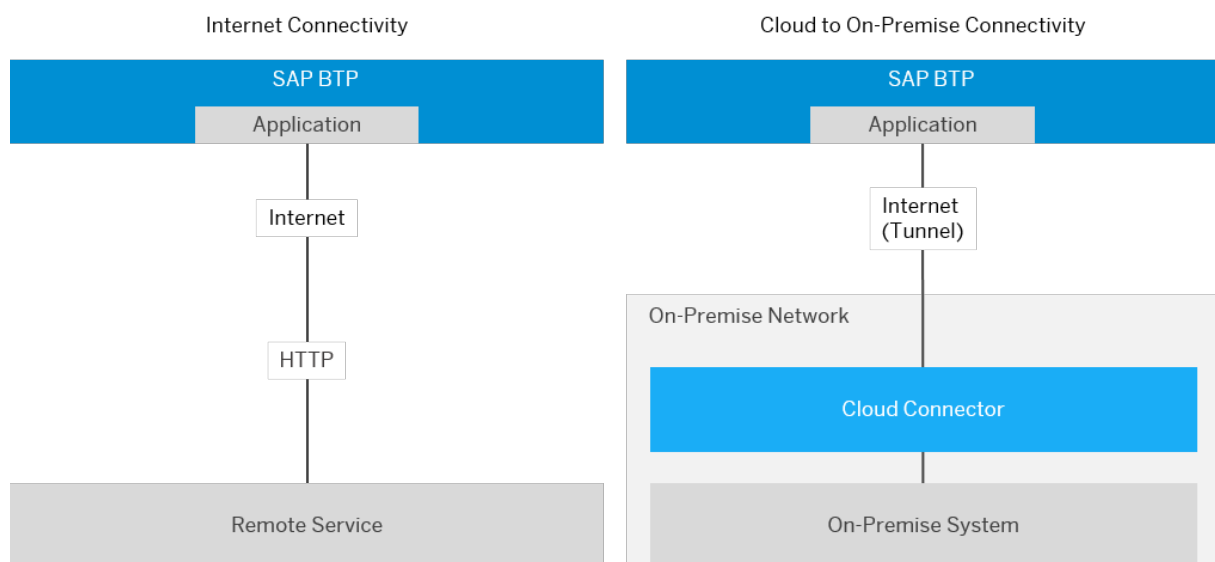
Support

- [Connectivity in the Cloud Foundry Environment \[page 8\]](#)
- [Cloud Connector \[page 343\]](#)
- [Connectivity Support \[page 869\]](#)

Overview

SAP BTP Connectivity allows SAP BTP applications to securely access remote services that run on the Internet or on-premise. This component:

- Allows subaccount-specific configuration of application connections via destinations.
- Provides a Java API that application developers can use to consume remote services.
- Allows you to make connections to on-premise systems, using the Cloud Connector.
- Lets you establish a secure tunnel from your on-premise network to applications on SAP BTP, while you keep full control and auditability of what is exposed to the cloud.
- Supports both the Neo and the Cloud Foundry environment for application development on SAP BTP.



A typical scenario for connecting your on-premise network to SAP BTP looks like this:

- Your company owns a global account on SAP BTP and one or more subaccounts that are assigned to this global account.
- Using SAP BTP, you subscribe to or deploy your own applications.
- To connect to these applications from your on-premise network, the Cloud Connector administrator sets up a secure tunnel to your company's subaccount on SAP BTP.
- The platform ensures that the tunnel can only be used by applications that are assigned to your subaccount.
- Applications assigned to other (sub)accounts cannot access the tunnel. It is encrypted via transport layer security (TLS), which guarantees connection privacy.

For inbound connections (calling an application or service on SAP BTP from an external source), you can use Cloud Connector [service channels \[page 608\]](#) (on-premise connections) or the respective API endpoints of your SAP BTP [region](#) (Internet connections).

Back to [Content \[page 4\]](#)

Features

SAP BTP Connectivity supports the following protocols and scenarios:

Protocol	Scenario
HTTP(S)	<p>Exchange data between your cloud application and Internet services or on-premise systems.</p> <ul style="list-style-type: none">• Create and configure HTTP destinations to make Web connections.• Connect to on-premise systems via HTTP, using the Cloud Connector.
RFC	<p>Invoke on-premise ABAP function modules via RFC.</p> <ul style="list-style-type: none">• Create and configure RFC destinations.• Make connections to back-end systems via RFC, using the Cloud Connector.
TCP	<p>Access on-premise systems via TCP-based protocols using a SOCKS5 proxy.</p>

Back to [Content \[page 4\]](#)

Restrictions

[General \[page 6\]](#)

[Protocols \[page 7\]](#)

[Cloud Foundry Environment \[page 7\]](#)

[Cloud Connector \[page 8\]](#)

ⓘ Note

For information about general SAP BTP restrictions, see [Prerequisites and Restrictions](#).

General

Topic	Restriction
Java Connector	<p>To develop a Java Connector (JCo) application for RFC communication, your SDK local runtime must be hosted by a 64-bit JVM, on a x86_64 operating system (Microsoft Windows OS, Linux OS, or Mac OS X).</p> <p>On Windows platforms, you must install the Microsoft Visual Studio C++ 2013 runtime libraries (vcredist_x64.exe), see Visual C++ Redistributable Packages for Visual Studio 2013 .</p>
Ports	For Internet connections, you are allowed to use any port > 1024 . For cloud to on-premise solutions there are no port limitations.
Destination Configuration	<ul style="list-style-type: none"> You can use destination configuration files with extension .props, .properties, .jks, and .txt, as well as files with no extension. If a destination configuration consists of a keystore or truststore, it must be stored in JKS files with a standard .jks extension.

Back to [Restrictions \[page 6\]](#)

Protocols

For the cloud to on-premise connectivity scenario, the following protocols are currently supported:

Protocol	Info
HTTP	HTTPS is not needed, since the tunnel used by the Cloud Connector is TLS-encrypted.
RFC	You can communicate with SAP systems down to SAP R/3 release 4.6C. Supported runtime environment is SAP Java Buildpack with a minimal version of 1.8.0.
TCP	You can use TCP-based communication for any client that supports SOCKS5 proxies.

Back to [Restrictions \[page 6\]](#)

Cloud Foundry Environment

Topic	Restriction
Service Channels	Service channels are supported only for SAP HANA database, see Using Service Channels [page 608] .
E-Mail	E-mail functions are not supported.

Back to [Restrictions \[page 6\]](#)

Cloud Connector

Topic	Restriction
Scenarios	To learn in which system landscapes you can set up the Cloud Connector, see Extended Scenarios [page 348] .
Installation	To check all software and hardware restrictions for working with the Cloud Connector, see Prerequisites [page 351] .

Back to [Restrictions \[page 6\]](#)

Back to [Content \[page 4\]](#)

Related Information

[Connectivity in the Cloud Foundry Environment \[page 8\]](#)

[Cloud Connector \[page 343\]](#)

[Connectivity via Reverse Proxy \[page 867\]](#)

[Connectivity Support \[page 869\]](#)

[Connectivity Proxy for Kubernetes \[page 732\]](#)

[Transparent Proxy for Kubernetes \[page 800\]](#)

1.1 Connectivity in the Cloud Foundry Environment

Consuming SAP BTP Connectivity for your application in the Cloud Foundry environment: Overview.

Note

This documentation refers to SAP BTP, Cloud Foundry environment. If you are looking for information about the Neo environment, see [Connectivity for the Neo Environment](#).

Hover over the elements for a description. Click an element for more information.

[What is SAP BTP Connectivity?](#)

[What's New for Connectivity](#)

[Initial Setup](#)

[Developing Applications](#)

[Security](#)

[Monitoring and Troubleshooting](#)

- [What Is SAP BTP Connectivity? \[page 9\]](#)
- [What's New for Connectivity \[page 16\]](#)
- [Administration \[page 58\]](#)
- [Monitoring and Troubleshooting \[page 335\]](#)
- [Security \[page 335\]](#)
- [Developing Applications \[page 214\]](#)

Related Information

[Resilience Recommendations \[page 337\]](#)

[On-Premise Connectivity in the Kyma Environment \[page 339\]](#)

1.1.1 What Is SAP BTP Connectivity?

Use SAP BTP Connectivity for your application in the Cloud Foundry environment: available services, connectivity scenarios, user roles.

SAP BTP Connectivity lets you connect your SAP BTP applications to other Internet resources, or to your on-premise systems running in isolated networks. It provides an extensive set of features to choose different connection types and authentication methods. Using its configuration options, you can tailor access exactly to your needs.

Note

You can use SAP BTP Connectivity for the Cloud Foundry environment and for the Neo environment. This documentation refers to SAP BTP, Cloud Foundry environment. If you are looking for information about the Neo environment, see [Connectivity for the Neo Environment](#).

Content

Hover over the elements for a description. Click an element for more information.

Features

Use Cases

User Roles

- [Features \[page 10\]](#)
- [Use Cases \[page 10\]](#)
- [User Roles \[page 11\]](#)

Features

SAP BTP Connectivity provides two services for the Cloud Foundry environment, the Connectivity service and the Destination service.

The Destination service and the Connectivity service together provide virtually the same functionality that is included in the Connectivity service of the Neo environment.

In the Cloud Foundry environment however, this functionality is split into two separate services:

- The **Connectivity** service provides a connectivity proxy that you can use to access on-premise resources.
- Using the **Destination** service, you can retrieve and store the technical information about the target resource (destination) that you need to connect your application to a remote service or system.

You can use both services together as well as separately, depending on the needs of your specific scenario.

Back to [Content \[page 9\]](#)

Use Cases

- Use the **Connectivity** service to connect your application or an SAP HANA database to *on-premise systems*:
 - Set up on-premise communication via HTTP or RFC for your cloud application.
 - Use a service channel to connect to an SAP HANA database on SAP BTP from your on-premise system, see [Configure a Service Channel for an SAP HANA Database](#).
- Use the **Destination** service:
 - To retrieve technical information about destinations that are required to consume the Connectivity service (optional), *or*
 - To provide destination information for connecting your Cloud Foundry application to any other *Web application* (remote service). This scenario does not require the Connectivity service.

Back to [Content \[page 9\]](#)

User Roles

In this document, we refer to different types of user roles – *responsibility roles* and *technical roles*. Responsibility roles describe the required user groups and their general tasks in the end-to-end setup process. Configuring technical roles, you can control access to the dedicated cloud management tools by assigning specific permissions to users.

[Responsibility Roles \[page 11\]](#)

[Technical Roles \[page 11\]](#)

Responsibility Roles

The end-to-end use of the Connectivity service and the Destination service requires these **user groups**:

- *Application operators* - are responsible for productive deployment and operation of an application on SAP BTP. Application operators are also responsible for configuring the remote connections (destination and trust management) that an application might need, see [Administration \[page 58\]](#).
- *Application developers* - develop a connectivity-enabled SAP BTP application by consuming the Connectivity service and/or the Destination service, see [Developing Applications \[page 214\]](#).
- *IT administrators* - set up the connectivity to SAP BTP in your on-premise network, using the [Cloud Connector \[page 343\]](#).

Some procedures on the SAP BTP can be done by developers as well as by application operators. Others may include a mix of development and operation tasks. These procedures are labeled using icons for the respective task type.

Task Types



Operator



Developer



Operator and/or Developer

Technical Roles

To perform connectivity tasks in the Cloud Foundry environment, the following **technical roles** apply:

[Technical Roles \[Feature Set A\] \[page 11\]](#)

[Technical Roles \[Feature Set B\] \[page 12\]](#)

Note

To apply the correct technical roles, you must know on which cloud management tools feature set (A or B) your account is running. For more information on feature sets, see [Cloud Management Tools – Feature Set Overview](#).

Technical Roles [Feature Set A]

Technical Connectivity Roles and Operations [Feature Set A]

Level	Operation (SAP BTP Cockpit or Cloud Connector)	Role
Subaccount	Connect a Cloud Connector to a subaccount (Cloud Connector)	One of these roles: <ul style="list-style-type: none"> • <i>Global Account</i> member See Add Members to Your Global Account. • <i>Security Administrator</i> (must be Global Account member or Cloud Foundry Org/Space member) See Managing Security Administrators in Your Subaccount [Feature Set A].
	Disconnect a Cloud Connector (cockpit)	
	Manage destinations (all CRUD operations) on subaccount level (cockpit)	
	View destinations (read operations) on subaccount level (cockpit)	
	Manage certificates (all CRUD operations) on subaccount level (cockpit)	
	View certificates (read operations) on subaccount level (cockpit)	
	Generate or renew the subaccount key pair for trust management (cockpit)	
Subaccount	View Cloud Connectors connected to a subaccount (cockpit)	A Cloud Foundry org role containing the permission <code>readSCCTunnels</code> , for example, the role <code>Org Manager</code> .
Service instance	Manage destinations (all CRUD operations) on service instance level (cockpit)	One of these roles: <ul style="list-style-type: none"> • <i>Org Manager</i> • <i>Space Manager</i> • <i>Space Developer</i> See User and Member Management .
	View destinations (read operations) on service instance level (cockpit)	
	Manage certificates (all CRUD operations) on service instance level (cockpit)	
	View certificates (read operations) on service instance level (cockpit)	

Note

As a prerequisite, a Cloud Foundry org must be available.

Back to [Technical Roles \[page 11\]](#)

Back to [User Roles \[page 11\]](#)

Technical Roles [Feature Set B]

Feature set B provides **dedicated roles** for specific operations. They can be assigned to **custom role collections**, but some of them are also available in **default role collections**.

Note

To see the Destination editor on subaccount level, you must have at least the *Destination Viewer* role, or both the *Destination Configuration Viewer* and the *Destination Certificate Viewer* roles.

For the Destination editor on service instance level, the corresponding roles apply: You need at least the *Destination Viewer Instance* role, or both the *Destination Configuration Instance Viewer* and the *Destination Certificate Instance Viewer* roles.

Technical Connectivity Roles and Operations [Feature Set B]

Level	Operation (SAP BTP Cockpit or Cloud Connector)	Role
Subaccount	Connect a Cloud Connector to a sub-account (Cloud Connector)	Cloud Connector Administrator
	Manage destinations (all CRUD operations) on subaccount level (cockpit)	One of these roles: <ul style="list-style-type: none"> • Destination Administrator • Destination Configuration Administrator
	View destinations (read operations) on subaccount level (cockpit)	One of these roles: <ul style="list-style-type: none"> • Destination Viewer • Destination Configuration Viewer
	Manage certificates (all CRUD operations) on subaccount level (cockpit)	One of these roles: <ul style="list-style-type: none"> • Destination Administrator • Destination Certificate Administrator
	View certificates (read operations) on subaccount level (cockpit)	One of these roles: <ul style="list-style-type: none"> • Destination Viewer • Destination Certificate Viewer
	Generate or renew the subaccount key pair for trust management (cockpit)	One of these roles: <ul style="list-style-type: none"> • Destination Administrator • Destination Subaccount Trust Administrator

Level	Operation (SAP BTP Cockpit or Cloud Connector)	Role
	Download the subaccount key pair for trust management (cockpit)	One of these roles: <ul style="list-style-type: none"> • Destination Viewer • Destination Subaccount Trust Viewer
Subaccount	View Cloud Connectors connected to a subaccount (cockpit)	A role containing the permission <code>readSCCTunnels</code> , for example, the predefined role <code>Cloud Connector Administrator</code> .
Service instance	Manage destinations (all CRUD operations) on service instance level (cockpit)	One of these roles: <ul style="list-style-type: none"> • Destination Administrator • Destination Configuration Administrator • Destination Administrator Instance • Destination Configuration Instance Administrator <p><i>plus</i> one of these roles:</p> <ul style="list-style-type: none"> • <i>Org Manager</i> • <i>Space Manager</i> • <i>Space Developer</i> <p>See User and Member Management.</p>
	View destinations (read operations) on service instance level (cockpit)	One of these roles: <ul style="list-style-type: none"> • Destination Viewer • Destination Configuration Viewer • Destination Viewer Instance • Destination Configuration Instance Viewer <p><i>plus</i> one of these roles:</p> <ul style="list-style-type: none"> • <i>Org Manager</i> • <i>Space Manager</i> • <i>Space Developer</i> <p>See User and Member Management.</p>

Level	Operation (SAP BTP Cockpit or Cloud Connector)	Role
	Manage certificates (all CRUD operations) on service instance level (cockpit)	<p>One of these roles:</p> <ul style="list-style-type: none"> • Destination Administrator • Destination Certificate Administrator • Destination Administrator Instance • Destination Certificate Instance Administrator <p><i>plus</i> one of these roles:</p> <ul style="list-style-type: none"> • <i>Org Manager</i> • <i>Space Manager</i> • <i>Space Developer</i> <p>See User and Member Management.</p>
	View certificates (read operations) on service instance level (cockpit)	<p>One of these roles:</p> <ul style="list-style-type: none"> • Destination Viewer • Destination Certificate Viewer • Destination Viewer Instance • Destination Certificate Instance Viewer <p><i>plus</i> one of these roles:</p> <ul style="list-style-type: none"> • <i>Org Manager</i> • <i>Space Manager</i> • <i>Space Developer</i> <p>See User and Member Management.</p>

Back to [Technical Roles \[Feature Set B\] \[page 12\]](#)

Default Role Collections [Feature Set B]

Default Role Collection	Connectivity Roles Included
Subaccount Administrator	<ul style="list-style-type: none"> • Cloud Connector Administrator • Destination Administrator
Subaccount Viewer	<ul style="list-style-type: none"> • Cloud Connector Auditor • Destination Viewer
Cloud Connector Administrator	Cloud Connector Administrator
Destination Administrator	Destination Administrator

Default Role Collection	Connectivity Roles Included
Connectivity and Destination Administrator	<ul style="list-style-type: none"> • Cloud Connector Administrator • Destination Administrator

Note

You can access subaccount-level destinations in two ways:

- Via the cockpit (as described above)
- Via the Destination service [REST API](#)

If a user has access to the Destination service REST API (via service instance binding credentials or a service key), he has full access to the destination and certificate configurations managed by that instance of the Destination service.

For more information, see [About Roles in the Cloud Foundry Environment](#) and check the activity *Instantiate and bind services to apps* in the linked Cloud Foundry documentation (docs.cloudfoundry.org).

Additionally, applications have access to the REST API of the Destination service instance they are bound to.

Back to [Technical Roles \[Feature Set B\] \[page 12\]](#)

Back to [Technical Roles \[page 11\]](#)

Back to [User Roles \[page 11\]](#)

Back to [Content \[page 9\]](#)

Related Information

[What's New for Connectivity \[page 16\]](#)

[Administration \[page 58\]](#)

[Developing Applications \[page 214\]](#)

[Security \[page 335\]](#)

[Monitoring and Troubleshooting \[page 335\]](#)

[Security Administration: Managing Authentication and Authorization](#)

1.1.2 What's New for Connectivity

Find the latest features, enhancements and bug fixes for SAP BTP Connectivity .

[What's New for Connectivity](#)

Related Information

- [2021 Connectivity \(Archive\) \[page 17\]](#)
- [2020 Connectivity \(Archive\) \[page 34\]](#)
- [2019 Connectivity \(Archive\) \[page 43\]](#)
- [2018 Connectivity \(Archive\) \[page 50\]](#)
- [2017 Connectivity \(Archive\) \[page 54\]](#)

1.1.2.1 2021 Connectivity (Archive)

2021

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Cloud Connector - SAP Cloud PKI	<p>SAP Cloud PKI (public key infrastructure) is enabled for technical communication between Cloud Connector and Connectivity service.</p> <ul style="list-style-type: none"> • The change is transparent for the Cloud Connector - as soon as you renew your subaccount certificate in the Cloud Connector, the newly issued X.509 client certificate will be part of SAP Cloud PKI. • If you are using Connectivity proxy software components as part of your solution, make sure you use version 2.4.1 or higher. For scenarios with termination in the ingress, version 2.3.1 of the Connectivity proxy is sufficient. 	Info only	New	2021-12-02
Connectivity	Integration Suite	Cloud Foundry Neo	Destination Service - Mail Destinations	<p>You can now configure destinations of type MAIL with any OAuth-based authentication type as available for HTTP destinations, including the option for mTLS via X.509 client certificate.</p>	Info only	New	2021-12-02

⚠ Restriction

The Mail Java API (Neo environment) does not provide the `javax.mail.Session` object out of the box. It must be configured manually.

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry Neo	Cloud Connector 2.14.0.1 - Enhancements	<p>Release of Cloud Connector version 2.14.0.1 introduces the following enhancements:</p> <ul style="list-style-type: none"> • Cloud Connector can now use SAPMachine 11 as Java runtime. For more information, see Prerequisites. • Cloud Connector supports Windows Server 2022 as additional OS version. For more information, see Prerequisites. • Monitoring was extended to show usage information for service channels (on-premise to cloud scenarios). For more information, see Monitoring. • An administrator can now configure more connectivity-related parameters on the Configuration > Advanced screen instead of modifying the configuration files on OS level. For more information, see Configure Tunnel Connections. • You can define a different location for audit log and trace files. For more information, see Manage Audit Logs and Troubleshooting • Additional configuration REST APIs let you configure the Cloud Connector remotely. For more information, see Configuration REST APIs. • The additional role <i>Subaccount Administrator</i> lets you define authorizations limited to subaccount-related tasks. Cross-subaccount configuration can only be viewed by users having this role. For more information, see Use LDAP for Authentication. • Access control usage monitor data is now persisted and will survive a restart. • The connection check for HTTPS access control entries now reveals information about the causes for a failing check and potential configuration issues if principal propagation with x.509 certificates is used. 	Recommended	New	2021-12-02

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
				<p>Action: We recommend that you always use the latest Cloud Connector version.</p> <p>For more information, see Upgrade.</p>			
Connectivity	Integration Suite	Cloud Foundry Neo	Cloud Connector 2.14.0.1 - Fixes	<p>Release of Cloud Connector version 2.14.0.1 provides the following bug fixes:</p> <ul style="list-style-type: none"> • Audit log selection was not working correctly if the end of the time interval was before noon. This issue has been fixed. • If an RFC connection is broken while waiting for data from the ABAP system, the processing engine could get into an inconsistent state, causing wrong processing for succeeding requests sent from the cloud application, which eventually could make the cloud application hang. This issue has been fixed. • Fixed a race condition that could occur if many requests were sent from the cloud application over the same RFC connection, and if the network from the cloud application to the Cloud Connector was very fast. In such a situation, two threads were processing this single RFC connection, causing an inconsistency that could lead to a <code>NullPointerException</code> in <code>RfcBlock.populateRequestStatistics</code> when trying to access the field <code><performanceStatistics></code>. • When configuring an RFC SNC access control entry, but overall SNC configuration is incomplete, Cloud Connector now reports the configuration error at runtime instead of falling back to plain RFC, if offered by the backend. 	Info only	Changed	2021-12-02

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Features	<ul style="list-style-type: none"> The Destinations UI in the cockpit lets you configure an X.509 client certificate for automatic token retrieval when using the relevant OAuth-based authentication types. The <code>AuthenticationHeaderProvider</code> Java client library, part of SAP Java Buildpack, was adapted as well. The REST API already supports it. The <code>ProxyType</code> attribute now offers the new option <code>PrivateLink</code>, allowing you to configure a destination with URL, and optionally a token service URL, pointing to services consumed via the SAP Private Link service (beta). For more information, see also What Is SAP Private Link Service (Beta)?. 	Info only	New	2021-11-18
				<div style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 10px;"> <p>Note</p> <p>The <code>CheckConnection</code> functionality as well as automatic token retrieval are not yet supported.</p> </div>			
Connectivity	Integration Suite	Neo	Destination Service - Features	The Destinations UI in the cockpit lets you configure an X.509 client certificate for automatic token retrieval when using the relevant OAuth-based authentication types. The <code>AuthenticationHeaderProvider</code> Java client library, part of the Neo runtimes, was adapted as well.	Info only	New	2021-11-18
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fix	<ul style="list-style-type: none"> A change has been applied that improves the stability and availability of the service during startup, for example, in case of a rolling update. A request processing change has been applied, improving asynchronous handling of the processing load, ultimately improving overall service stability and availability. 	Info only	Changed	2021-11-18

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Destination Service - OAuth	<ul style="list-style-type: none"> Destinations with <code>ProxyType</code> set to <code>OnPremise</code> can now be configured with OAuth-based authentication types, both via the BTP cockpit UI and the Destination service REST API. Automated token retrieval from OAuth servers residing on premise, exposed via Connectivity service and Cloud Connector, is now supported. 	Info only	New	2021-11-04
Connectivity	Integration Suite	Cloud Foundry Kyma	Connectivity Proxy Version 2.4.1	<p>Connectivity proxy version 2.4.1 is now available.</p> <ul style="list-style-type: none"> Connectivity CA is now automatically downloaded during help deployment. Applies critical preparation for adoption of SAP Cloud PKI. Improved server certificate validation towards remote targets. Multiple open source software components reported as vulnerable have been replaced. 	Info only	Announcement	2021-10-21
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Bug Fix	<p>An internal service exception could result in a <i>502 Bad Gateway</i> error on the client side, which was visible in the Cloud Connector logs during automatic reconnect.</p> <p>This issue has been fixed.</p>		Changed	2021-10-07
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fixes	<ul style="list-style-type: none"> In some cases, a Destination service instance could not be deleted. The operation now works as expected. A performance optimisation reduces the overall amount of remote calls to XSUAA when the Destination service is called with a user token. As a result, less load is put on XSUAA, and the Destination service responds faster. This fix contributes to improving overall stability. 		Changed	2021-10-07
Connectivity	Integration Suite	Cloud Foundry	SAP Java Buildpack	<p>SAP Java Buildpack has been updated from version 1.38.0 to 1.39.0.</p> <ul style="list-style-type: none"> The <code>com.sap.cloud.security.xsuaa</code> API has been updated to version 2.10.5. The Connectivity API extension has been updated to version 3.12.0. 		New	2021-09-13

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Authentication Types	<p>When using authentication type <code>OAuth2ClientCredentials</code>, you can choose a tenant to perform automated token retrieval that is different from the tenant used to look up the destination configuration.</p> <p>This feature is especially useful for automated service scenarios, like running offline jobs, and so on.</p> <div style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 10px; margin: 10px 0;"> <p>Note</p> <p>You cannot use this feature in combination with a passed user context. In this case, the tenant used to perform automated token retrieval is exclusively determined by the user context.</p> </div> <p>For more information, see OAuth Client Credentials Authentication.</p>		New	2021-08-26
Connectivity	Integration Suite	Neo	Destinations - Timeout Properties	<p>You can configure timeout properties in the destination configuration, following a documented naming convention.</p> <p>This feature lets you manage timeouts externally, regardless of the cloud application's lifecycle.</p> <p>Using HttpDestination library version 2.15, timeout properties are processed at runtime when pre-configuring the HTTP client instance for the cloud application.</p> <p>For more information, see HTTP Destinations.</p>		New	2021-08-26
Connectivity	Integration Suite	Cloud Foundry Neo	Cloud Connector - Security Fixes	<p>Multiple vulnerabilities in the Cloud Connector have been fixed.</p> <p>For more information, see SAP security note 3058553.</p>		Changed	2021-08-12
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Client Certificates	<p>When generating an X.509 client certificate (as announced on July 1), you can set a password to protect the private key. If you choose a PKCS12 file format, also the keystore is protected by the same password.</p>		New	2021-08-12

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Neo	Destination Service - HTTP Headers	As of HttpDestination version 2.14.0 , any defined HTTP header in the destination configuration (see HTTP Destinations) is processed at runtime, that is, it is added to the request that is sent to the target server.		New	2021-08-12
Connectivity	Integration Suite	Neo	Connectivity Service - Bug Fix	A few stabilisation fixes have been applied in the Connectivity service to handle rare cases in which an abnormal amount of metering data was received by the Cloud Connector. This could cause a partial blockage of the service.		Changed	2021-08-12
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fix	A few stabilisation fixes have been applied on the REST server-side logic, allowing more efficient parallel request handling under load.		Changed	2021-08-12
Connectivity	Integration Suite	Cloud Foundry Neo	Cloud Connector 2.13.2 - Enhancements	Release of Cloud Connector version 2.13.2 introduces the following improvement: <ul style="list-style-type: none"> The HTML validation of login information has been improved. 		New	2021-07-15

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry Neo	Cloud Connector 2.13.2 - Fixes	<p>Release of Cloud Connector version 2.13.2 provides the following bug fixes:</p> <ul style="list-style-type: none"> A regression prevented that links provided in the login info widget (login screen) could be clicked. This issue has been fixed. When rewriting a location header for redirect responses (status codes 30x), the lookup to determine the virtual host that replaces the internal one is now case insensitive. When using custom attributes in JWTs (JSON web tokens) for principal propagation, the value can now be extracted even if represented as single element array. A slow network could prevent a successful initial push, caused by an unintended timeout. As a consequence, the configuration on the shadow instance could be incomplete, even though the shadow showed a successful connection. This issue has been fixed. CPIC traces can now be turned on and off multiple times without the need to restart. Issues with restoring a 2.13.x backup into a fresh installation on Linux have been fixed. 		Changed	2021-07-15

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Destination Service	<ul style="list-style-type: none"> Using the Destination service REST API, you can configure a service-side issued X.509 client certificate as part of the SAP Cloud PKI (public key infrastructure) and formally choose <i>automatic renewal</i> of the certificate. The same feature is available in the <i>Destinations</i> editor of the cloud cockpit (Connectivity > Destinations). The <i>SAP Java Buildpack</i> now includes Java APIs as part of a client library for the Destination service. You can use the <code>ConnectivityConfiguration</code> Java API to retrieve destination and certificate configurations, and <code>AuthenticationHeaderProvider</code> Java API to provide prepared HTTP headers holding authentication tokens for various scenarios. For more information, see Destination Java APIs. 		New	2021-07-01
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Destinations Editor	<p>The <i>Destinations</i> UI (editor) in the cockpit lets you create a certificate configuration entry containing an X.509 client certificate part of SAP Cloud PKI (public key infrastructure).</p> <p>Optionally, you can specify values for the certificate common name (CN) as well as for the validity period (minimum value: one day, maximum value: one year). This feature has already been available via the Destination service REST API.</p>		New	2021-06-17
Connectivity	Integration Suite	Cloud Foundry Neo	Destination Service - Destinations Editor	<p>The <i>Destinations</i> UI (editor) in the cockpit has introduced a warning message as a reminder that the user's personal password should not be used when configuring the authentication type of a destination, for example, <code>BasicAuthentication</code> or <code>OAuth2Password</code>.</p> <p>The reason behind is that by design, destination configurations are meant to be used by one or more cloud applications which typically are used by more than one person.</p>		New	2021-06-17

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry Neo	Connectivity Service - Bug Fix	In rare cases, the Cloud Connector version shown in the cockpit (Connectivity → Cloud Connectors) got lost. This issue has been fixed.		Changed	2021-06-17
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fix	An issue has been resolved which prevented updating a service instance (previously created in the cockpit) via the Cloud Foundry CLI.		Changed	2021-06-17
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Principal Propagation	You can configure a principal propagation scenario using a user access token issued by the Identity Authentication service (IAS), in addition to the scenarios based on XSUAA. For more information, see Principal Propagation via IAS Token .		New	2021-06-03
Connectivity	Integration Suite	Cloud Foundry	Destination Service - HTTP Destinations	A naming convention has been introduced, specifying how to properly configure HTTP headers and queries in a destination configuration. For more information, see HTTP Destinations .		New	2021-06-03
Connectivity	Integration Suite	Cloud Foundry Neo	Connectivity Service - Tunnel Connections	On the cloud side, the tunnel connection idle threshold has been increased to better match both older (yet supported) and latest Cloud Connector versions (versions lower or equal to 2.13). This ensures the internal heartbeat mechanism would work properly even in some special cases in which short interruptions have been observed.		Changed	2021-06-03
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bugfix	The service could have experienced delays in case of parallel load which caused requests to be executed unexpectedly slower on random basis. This issue has been fixed.		Changed	2021-06-03

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry Neo	Java Connector 3.1.4.0 - Enhancements	<p>Release of Java Connector (JCo) version 3.1.4.0 introduces the following features and improvements:</p> <ul style="list-style-type: none"> JCo now offers the ABAP server processing time for JCo client scenarios via method <code>JCoThroughput.getServerTime()</code>. <code>JCoRepository</code> methods were enhanced to ignore trailing blanks in passed structure, table, and function module names which are supposed to be looked up. Performance was improved for setting <code>DATE</code> and <code>TIME</code> datatype fields when using <i>strings</i> as input values. 		New	2021-05-20

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry Neo	Java Connector 3.1.4.0 - Bug Fixes	<p>Release of Java Connector (JCo) version 3.1.4.0 provides the following bug fixes:</p> <ul style="list-style-type: none"> When invoking remote function modules (RFMs) via the t/q/bgRFC protocol to the same <code>JCoDestination</code> in multiple threads simultaneously, used TIDs, queue names and unit IDs could have been overwritten and used in the wrong thread context, which might have led to data loss in the target system. For example, IDocs that seemed to have been transferred correctly without an error, were not stored in the target system, because the TID contract was broken and several IDocs were erroneously sent at the same time with the same TID although different ones had been specified. This issue has been fixed. <div style="border: 1px solid #ccc; background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <p>Note</p> <p>This regression bug was introduced with JCo 3.1.3.</p> </div> <ul style="list-style-type: none"> When the new reentrance ticket technology, introduced as of S/4HANA 1909, was used to log on to the communication partner system, and <code>JCoCustomRepository</code> was configured to use query mode <code>DISABLE_REPOSITORY_POOL</code>, querying RFC metadata resulted in a logon failure (<i>invalid logon ticket</i>). This issue has been fixed. If a JSON document contained numeric fields with negative values, for example, for a field of type <code>BCD</code> or <code>INT</code>, the JSON parser did not accept the sign character when analyzing the value for the given field. In this case, <code>JCoRecord.fromJSON()</code> threw an exception similar to <code>JCoSerializationException: (191) JCO_ERROR_SERIALIZATION: Digit(s) expected near position <###></code>. This issue has been fixed. 	Changed		2021-05-20

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Cockpit: Cloud Connectors View	The Cloud Connectors view in the SAP BTP cockpit is now also available if your account is running on cloud management tools feature set B. For more information, see Monitoring (section <i>Monitoring from the Cockpit</i>).		New	2021-04-22
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.13.1 - Enhancements	<p>Release of Cloud Connector version 2.13.1 introduces the following features and improvements:</p> <ul style="list-style-type: none"> • Additional audit log entries for changing the trace level are available. • You can open the support log assistant directly from the <i>Log And Trace Files</i> screen. For more information, see Troubleshooting, section <i>Log And Trace Files</i>. • The dependency on ping checks for connections to the LDAP system, which is used for UI authentication, has been minimized to avoid unnecessary role switches in high availability mode. 		New	2021-03-25

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.13.1 - Fixes	<p>Release of Cloud Connector version 2.13.1 provides the following bug fixes:</p> <ul style="list-style-type: none"> Connecting a subaccount with several hundred access control entries is working again. With release 2.13.1, restoring a backup created in version 2.13.0 works properly again for a Cloud Connector running on Linux. Backups created in version 2.12.5 and older can be restored properly. Failures on restore led to a non-usable Cloud Connector setup. The following high availability issues have been fixed: <ul style="list-style-type: none"> Improved implementation ensures that a high availability setup does not end up in a shadow/shadow situation. This issue could occur under rare circumstances. Errors could occur if subaccounts have a larger number of access control entries. Network issues could prevent the individual replication of configuration changes. After switching roles, connections can now always be reestablished correctly. The connection test for LDAPS access control entries now works correctly. A memory leak in the comprised netty library has been fixed by upgrading to a newer version. A subaccount display issue has been fixed: In version 2.13.0, subaccounts on <i>eu2.hana.ondemand.com</i> were displayed as belonging to region Europe (Rot) instead of Europe (Frankfurt). 	Changed		2021-03-25
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Mobile Service Instances	You can create a destination configuration pointing to a mobile service instance, resulting in a fully functional destination configuration, including automatic token retrieval for the respective OAuth flows supported by the mobile service.	New		2021-02-25

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Kyma Other	Destination Service - Consumption from Kyma or Kubernetes Environments	Consumption of the Destination service from the Kyma or Kubernetes environments has been officially documented. For more information, see Create and Bind a Destination Service Instance .		New	2021-02-25
Connectivity	Integration Suite	Cloud Foundry	Principal Propagation Authentication - OpenID Connect	When sending a user principal via the HTTP header <code>X-user-token</code> , you can use any <i>OpenID Connect</i> -compliant OAuth server and a related OpenID access token for passing the user identity. To enable this feature, you must specify either <code>x_user_token . jwks_uri</code> or <code>x_user_token . jwks</code> as additional attribute, as described in the respective authentication type. or For more information, see HTTP Destinations .		New	2021-02-11
Connectivity	Integration Suite	Cloud Foundry	OAuth - X.509 Client Certificates	You can use X.509 client certificates for OAuth flows supported by the respective authentication types, see OAuth with X.509 Client Certificates .		New	2021-02-11
Connectivity	Integration Suite	Cloud Foundry	"Find Destination" REST API - Skip Credentials	The "Find Destination" REST API endpoint has been enhanced with a new feature enabling the client application to initiate a skip of credentials in the returned response. This parameter is useful especially for OAuth destinations (such as <i>OAuth2 User Token Exchange</i> , <i>OAuth2 JWT Bearer</i> , <i>OAuth2 SAML Bearer Assertion</i>). The client application may actually need only the auto-retrieved token by the service, which makes the credentials optional for the application, and in certain cases they are preferred not to be returned in the response. For more information, see SAP API Business Hub .		New	2021-02-11

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Destinations - Authentication Types	<ul style="list-style-type: none"> The property <code>SystemUser</code> is deprecated. The cockpit now shows an alert if this feature is still in use, suggesting what to do instead. Alternatives for technical user authentication are <i>Basic Authentication</i>, <i>OAuth2 Client Credentials</i>, or <i>Client Certificate Authentication</i>. See also OAuth SAML Bearer Assertion Authentication. <div style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 10px; margin: 10px 0;"> <p>Note</p> <p>In general, we recommend that you work on behalf of specific (named) users rather than working with a technical user. To extend an OAuth access token's validity, consider using an OAuth refresh token.</p> </div> <ul style="list-style-type: none"> Authentication type <i>SAP Assertion SSO</i> is deprecated. The cockpit now shows an alert if this feature is still in use, suggesting what to do instead. Authentication types <i>Principal Propagation</i> (for on-premise connections), <i>OAuth2 SAML Bearer Assertion</i> (Internet connections) or <i>SAML Assertion</i> (Internet connections) are the recommended mechanisms for establishing single sign-on (SSO). See SAP Assertion SSO Authentication. 	Changed		2021-02-11
Connectivity	Integration Suite	Neo	Password Storage API	The Password Storage API documentation on SAP API Business Hub has been moved from the deprecated API package to SAP Cloud Platform Credential Store .	Changed		2021-01-28

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.13.0 - Enhancements	<p>Release of Cloud Connector version 2.13.0 introduces the following features and improvements:</p> <ul style="list-style-type: none"> Cloud Connector 2.13 is based on a different runtime container. Up to version 2.12.x, the JavaWeb 1.x runtime based on Tomcat 7 was used. It now switches to JavaWeb 3.x based on Tomcat 8.5. As a consequence, the internal structure has changed and works differently. The upgrade will adjust these changes as much as possible for versions 2.9 and higher. Linux on <i>ppc64 little endian</i> (ppc64le) is added as a supported platform for the Cloud Connector. For more information, see Prerequisites. A set of new configuration REST APIs has been added. For more information, see Configuration REST APIs. An additional screen in the subaccount-specific monitoring provides usage statistics of the various access control entries. Alternatively, you can access the same data using a new monitoring REST API. For more information, see Monitoring. For access control entries of type TCP, you can configure a port range instead of a single port. For more information, see Configure Access Control (TCP). You can configure a widget that shows information about the Cloud Connector on the login screen. For more information, see Configure Login Screen Information. Improved high availability communication supersedes applying SAP note 2915578. For new Cloud Connector versions, a notification and alert is shown to help you schedule the update. The Cloud Connector supports JSON Web tokens (JWTs) based on OpenID-Connect (OIDC) for principal propagation authentication (Cloud Foundry environment). 		New	2021-01-14

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
				<p>For more information, see Configure Principal Propagation via OIDC Token.</p> <ul style="list-style-type: none"> Client-side load balancing based on round-robin was introduced for Cloud Connector connections to SAP Cloud Platform to address its endpoints which are exposed on multiple IP addresses for high availability. Scenarios based on the HTTP header <i>Expect: 100-continue</i> and response code <i>HTTP 100</i> are now supported. 			
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.13.0 - Fixes	<p>Release of Cloud Connector version 2.13.0 provides the following bug fixes:</p> <ul style="list-style-type: none"> When doing a rollover at midnight, the initial audit log entry for a new file was not created correctly and the audit log checker wrongly assessed such files as corrupted. This issue has been fixed. Incorrect host information could be used in audit logs related to access control audit entries. This issue has been fixed. 		Changed	2021-01-14
Connectivity	Integration Suite	Cloud Foundry	Cloud Connector - Subaccount Configuration	<p>For a subaccount that uses a custom identity provider (IDP), you can choose this IDP for authentication instead of the (default) SAP ID service when configuring the subaccount in the Cloud Connector.</p> <p>For more information, see Use a Custom IDP for Subaccount Configuration.</p>		New	2021-01-14

1.1.2.2 2020 Connectivity (Archive)

2020

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Java Connector (JCo) - Client Certificates	<p>JCo provides the new property <code>jco.client.tls_client_certificate_logon</code> to support the usage of a TLS client certificate for logging on to an ABAP system via WebSocket RFC.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> User Logon Properties (Cloud Foundry environment) User Logon Properties (Neo environment) <p>For more information on WebSocket RFC, see also:</p> <ul style="list-style-type: none"> WebSocket RFC 	New	2020-12-17
Connectivity	Integration Suite	Cloud Foundry	HTTP Destinations - Authentication Types	<p>Authentication type <i>SAP Assertion SSO</i> is deprecated. It will soon be removed as a feature from the Destination service.</p> <p>Use Principal Propagation SSO Authentication instead, which is the recommended mechanism for establishing single sign-on (SSO).</p>	Deprecated	2020-12-17
Connectivity	Integration Suite	Neo	HTTP Destinations - Authentication Types	<p>Authentication type <i>SAP Assertion SSO</i> is deprecated.</p> <p>Use Principal Propagation SSO Authentication instead, which is the recommended mechanism for establishing single sign-on (SSO).</p>	Deprecated	2020-12-17
Connectivity	Integration Suite	Cloud Foundry	HTTP Destinations - Destination Properties	<p>The destination property <code>SystemUser</code> for the authentication types:</p> <ul style="list-style-type: none"> OAuth SAML Bearer Assertion Authentication SAP Assertion SSO Authentication <p>will be removed soon. More information on timelines and required actions will be published in the release notes at a later stage.</p> <p>See also:</p> <ul style="list-style-type: none"> OAuth SAML Bearer Assertion Authentication SAP Assertion SSO Authentication 	Announce	2020-12-03

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	JCo Runtime - Enhancement	JCo Runtime 3.1.3.0 introduces the following enhancement: If the backend is known to be new enough, JCo does not check for the existence of RFC_METADATA_GET, thus avoiding the need to provide additional authorizations for the repository user.	New	2020-11-05
Connectivity	Integration Suite	Neo Cloud Foundry	JCo Runtime - Bug Fix	JCo Runtime 3.1.3.0 provides the following bug fix: Up to JCo 3.1.2, the initial value for fields of type STRING and XSTRING was <i>null</i> . Since the initial value check in ABAP is different, JCo now behaves the same way and uses an empty string and an empty byte array, respectively.	Changed	2020-11-05
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Automatic Token Retrieval	The Destination service offers a new feature related to the automatic token retrieval functionality, which lets the destination administrator define HTTP headers and query parameters as additional configuration properties, used at runtime when requesting the token service to obtain an access token. See HTTP Destinations .	New	2020-11-05
Connectivity	Integration Suite	Cloud Foundry	Documentation - Principal Propagation Scenarios	The documentation of principal propagation (user propagation) scenarios provides improved information on the basic concept and guidance on how to set up different scenarios. See Principal Propagation .	Changed	2020-10-22
Connectivity	Integration Suite	Cloud Foundry	Cloud Connector 2.12.5 - Enhancements	Release of Cloud Connector version 2.12.5 introduces the following improvements: <ul style="list-style-type: none"> For principal propagation scenarios, custom attributes stored in <code>xs.user.attributes</code> of the JWT (JSON Web token) are now accessible for the subject pattern. See Configure a Subject Pattern for Principal Propagation. Improved resolving for DNS names with multiple IP addresses by adding randomness to the choice of the IP to use. This is relevant for many connectivity endpoints in SAP Cloud Platform, Cloud Foundry environment. 	New	2020-10-22

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.5 - Fixes	<p>Release of Cloud Connector version 2.12.5 provides the following bug fixes:</p> <ul style="list-style-type: none"> After actively performing a master-shadow switch for a disaster recovery subaccount, a zombie connection could cause a timeout of all application requests to on-premise systems. This issue has been fixed. When refreshing the subaccount certificate in an high availability setup, transferring the changed certificate to the shadow was not immediately triggered, and the updated certificate could get lost. This issue has been fixed. If many RFC connections were canceled at the same time, the Cloud Connector could crash in the native layer, causing the process to die. This issue has been fixed. The LDAP configuration test now supports all possible configuration parameters. 	Changed	2020-10-22
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Service Instances - Quota Management	<p>When using service plan "lite", quota management is no longer required for this service. From any subaccount you can consume the service using service instances without restrictions on the instance count.</p> <p>Previously, access to service plan "lite" has been granted via entitlement and quota management of the application runtime. It has now become an integral service offering of SAP Cloud Platform to simplify its usage.</p> <p>See also Create and Bind a Connectivity Service Instance.</p>	Changed	2020-10-08
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Service Instances - Quota Management	<p>When using service plan "lite", quota management is no longer required for this service. From any subaccount you can consume the service using service instances without restrictions on the instance count.</p> <p>Previously, access to service plan "lite" has been granted via entitlement and quota management of the application runtime. It has now become an integral service offering of SAP Cloud Platform to simplify its usage.</p> <p>See also Create and Bind a Destination Service Instance.</p>	Changed	2020-10-08

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	SAP Java Buildpack - Java Connector (JCo)	<p>The SAP Java Buildpack has been updated from 1.27.3. to 1.28.0.</p> <ul style="list-style-type: none"> TomEE Tomcat has been updated from 7.0.104 to 7.0.105. SAPJVM has been updated to 81.65.65. The <i>com.sap.cloud.security.xsuaa</i> API has been updated from 2.7.5 to 2.7.6. The SAP HANA driver has been updated from 2.5.49 to 2.5.52. JCo-corresponding libraries have been updated: <i>connectivity</i> to 3.3.3, <i>connectivity apiext</i> to 0.1.37. The activation process for the JCo component in the SAP Java Buildpack has been changed. Starting with this release, it is activated by setting the following environment variable: <code><USE_JCO=true></code>. <div style="border: 1px solid #0070c0; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>The previous activation process for the JCo component is deprecated and will expire after a transition period.</p> </div>	Changed	2020-09-24
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Error Handling	Error handling has been improved for updating service instances via the Cloud Foundry CLI and the cloud cockpit when providing the configuration JSON data.	Changed	2020-09-10
Connectivity	Integration Suite	Neo	Connectivity Service - Bug Fix	A synchronization issue has been fixed on cloud side that in very rare cases could lead to a <i>zombie</i> tunnel from the Cloud Connector to SAP Cloud Platform, which required to reconnect the Cloud Connector.	Changed	2020-09-10
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fix	During <i>Check Connection</i> processing of a destination with basic authentication, the Destination service now uses the user credentials for both the HTTP HEAD and HTTP GET requests to verify the connection on HTTP level.	Changed	2020-09-10
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fix	Using authentication type <code>OAuth2SAMLBearerAssertion</code> , an issue could occur when adding the user's SAML group attributes into the resulting SAML assertion that is sent to the target token service. This issue has been fixed.	Changed	2020-08-13

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Destination Service - REST API - Pagination Feature	The REST API pagination feature provides improved error handling in case of issues with the pagination, for example, if an invalid page number is provided.	Changed	2020-08-13
Connectivity	Integration Suite	Neo	HttpDestination Library - New Version	The <code>HttpDestination</code> v2 library has been officially released in the Maven Central Repository . It enables the usage in Tomcat and TomEE-based runtimes the same way as in the deprecated <i>JavaWeb</i> and <i>Java EE 6 Web Profile</i> runtimes. See also HttpDestination Library .	New	2020-07-30
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fix	An error handling issue has been fixed in the Destination service, which is related to the recently introduced <i>SAP Assertion SSO authentication</i> type. If a wrong input was provided, you can now see the error properly, and recover it.	Changed	2020-07-30
Connectivity	Integration Suite	Cloud Foundry	Destinations - Authentication Types	You can use authentication type <code>OAuth2JWTBearer</code> when configuring a Destination. It is a simplified version of the authentication type <code>OAuth2UserTokenExchange</code> and represents the official OAuth grant type for exchanging OAuth tokens. See HTTP Destinations .	New	2020-07-02
Connectivity	Integration Suite	Cloud Foundry	Destination Service - HTTP Header	The Destination service provides a prepared HTTP header that simplifies application and service development. See HTTP Destinations (code samples).	New	2020-07-02
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fix	A concurrency issue in the Destination service, related to parallel auth token retrieval in the token cache functionality, could result in partial request failures. This issue has been fixed.	Changed	2020-07-02
Connectivity	Integration Suite	Cloud Foundry	HTTP Destinations - Authentication Types	The Cloud Foundry environment supports <i>SAP Assertion SSO</i> as authentication type for configuring destinations in the Destination service. See HTTP Destinations .	New	2020-06-18

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Destination Service REST API	The "Find Destination" REST API now includes the scopes of the automatically retrieved access token in the response that is returned to the caller. See "Find Destination" Response Structure .	New	2020-06-04
Connectivity	Integration Suite	Cloud Foundry	Destinations for Service Instances	For subscription-based scenarios, you can use an automated procedure to create a destination that points to your service instance. See Managing Destinations .	New	2020-06-04
Connectivity	Integration Suite	Neo Cloud Foundry	Connectivity Service - Bug Fix	In rare cases, establishing a secure tunnel between Cloud Connector (version 2.12.3 or older) and the Connectivity service could cause an issue that requires to manually disconnect and connect the Cloud Connector. This issue has been fixed. The fix requires Cloud Connector version 2.12.4 or higher.	Changed	2020-05-21
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.4 - Features	Release of Cloud Connector version 2.12.4 introduces the following features and enhancements: <ul style="list-style-type: none"> You can activate the SSL trace in the Cloud Connector administration UI also for the shadow instance. 	New	2020-05-07
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.4 - Fixes	Release of Cloud Connector version 2.12.4 provides the following bug fixes: <ul style="list-style-type: none"> You can edit and delete domain mappings in the Cloud Connector administration UI correctly. The REST API does no longer return an empty configuration. REST API DELETE operations do not require setting a content-type application/json to function properly. If more than 2000 audit log entries match a selection, redefining the search and getting a shorter list now works as expected. A potential leak of HTTP backend connections has been closed. 	Changed	2020-05-07
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Bug Fix	A fix has been applied in the Connectivity service internal load balancers, enabling the sending of TCP keep-alive packets on client and server side. This change mainly affects SOCKS5-based communication scenarios.	Changed	2020-03-26

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Destination Service Instances	You can create a service instance specifying an update policy. This allows you to avoid name conflicts with existing destinations. See Create and Bind a Destination Service Instance .	New	2020-03-26
Connectivity	Integration Suite	Cloud Foundry	Cockpit - Destination Management	The Destinations editor in the cockpit is available for accounts running on the cloud management tools feature set B. See Managing Destinations .	New	2020-03-12
Connectivity	Integration Suite	Neo	Connectivity Service - Bug Fix	When creating or editing a destination with authentication type <code>OAuth2ClientCredentials</code> in the cockpit, the parameter <code>Audience</code> could not be added as additional property. This issue has been fixed.	Changed	2020-03-12
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.3 - Features	Release of Cloud Connector version 2.12.3 introduces the following features and enhancements: <ul style="list-style-type: none"> When using the SAP JVM as runtime, the thread dump includes additional information about currently executed RFC function modules. The hardware monitor includes a Java Heap history, showing the usage in the last 24 hours. If you are using the file <code>scc_daemon_extension.sh</code> to extend the daemon in a Linux installation, the content is included in the initialization section of the daemon. This lets you make custom extensions to the daemon that survive an upgrade. See Installation on Linux OS, section <i>Installer Scenario</i>. 	New	2020-02-27

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.3 - Fixes	<p>Release of Cloud Connector version 2.12.3 provides the following bug fixes:</p> <ul style="list-style-type: none"> When switching roles between master and shadow instance in a high availability setup, the switch is no longer blocked by active RFC function module invocations. A fix in the backend HTTP connection handling prevents issues when the backend tries to send the HTTP response before completely reading the HTTP request. When sending large amounts of data to an on-premise system, and using RFC with a network that provides large bandwidth, the Cloud Connector could fail with the error message <i>Received invalid block with negative size</i>. This issue has been fixed. The Cloud Connector admin UI now shows the correct user information for installed Cloud Connector instances in the <i>About</i> window. Fixes in the context of disaster recovery: <ul style="list-style-type: none"> The location ID is now handled properly when setting it <i>after</i> adding the recovery subaccount. Application trust settings and application-specific connections are applied in the disaster case. Principal propagation settings are applied in the disaster case 	Changed	2020-02-27
Connectivity	Integration Suite	Neo Cloud Foundry	JCo Runtime - WebSocket RFC	<p>The JCo runtime in SAP Cloud Platform lets you use WebSocket RFC (RFC over Internet) with ABAP servers as of S/4HANA (on-premise) version 1909. In the RFC destination configuration, this is reflected by new configuration properties and by the option to choose between different proxy types.</p> <p>See Target System Configuration (Cloud Foundry environment), or Target System Configuration (Neo environment).</p>	New	2020-02-13
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service for Trial Accounts - Bug Fix	<p>The Connectivity service is operational again for trial accounts. A change in the Cloud Foundry Core component caused the service not be accessible by applications hosted in DiegoCell that are dedicated for trial usage in a separate VPC (virtual private cloud) account. This issue has been fixed.</p>	Changed	2020-01-30

1.1.2.3 2019 Connectivity (Archive)

2019

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.2 - Features	<p>Release of Cloud Connector version 2.12.2 introduces the following features and enhancements:</p> <ul style="list-style-type: none"> You can turn on the TLS trace from the Cloud Connector administration UI instead of modifying the <code>props.ini</code> file on OS level. See Troubleshooting. The status of the used subaccount certificate is shown on the <i>Subaccount</i> overview page of the Cloud Connector administration UI, in addition to expiring certificates shown in the <i>Alerting</i> view. See Establish Connections to SAP Cloud Platform. 	New	2019-12-05
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.2 - Fixes	<p>Release of Cloud Connector version 2.12.2 provides the following bug fixes:</p> <ul style="list-style-type: none"> Subject values for certificates requiring escaping are treated correctly. Establishing a connection to the master is now possible when being logged on to the shadow with a user that has a space in its name. Performance statistics could show too long total execution times. This issue has been fixed. IP address changes for the connectivity service hosts are recognized properly. The Cloud Connector could crash on Windows, when trying to enable the payload trace with 4-eyes-principle without the required user permissions. This issue has been fixed. 	Changed	2019-12-05
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Bug Fix	<p>Applications sending a significant amount of data payload during OAuth authorization processing could cause an out-of-memory error on the Connectivity service side. This issue has been fixed.</p>	Changed	2019-11-21

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo	Region Europe (Frankfurt) - Change of Connectivity Service Hosts	<p>The following IP addresses of the Connectivity service hosts for region Europe/Frankfurt (eu2.hana.ondemand.com) will change on 26 October 2019:</p> <ul style="list-style-type: none"> connectivitynotification.eu2.hana.ondemand.com: from 157.133.70.140 (current) to 157.133.206.143 (new) connectivitycertsigning.eu2.hana.ondemand.com: from 157.133.70.132 (current) to 157.133.205.174 (new) connectivitytunnel.eu2.hana.ondemand.com: from 157.133.70.141 (current) to 157.133.205.233 (new) <p>If you have allowed the current addresses or IP ranges in your firewall rules, make sure you also include the new values before 26 October 2019.</p> <p>See also: Prerequisites: Network.</p>	Announcement	2019-10-03
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Connection Check	<p>Using the <i>Destinations</i> editor in the cockpit, you can check connections also for on-premise destinations.</p> <p>See Check the Availability of a Destination.</p>	Changed	2019-09-26
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector - Java Runtime	<p>The support for using Cloud Connector with Java runtime version 7 will end on December 31, 2019. Any Cloud Connector version released after that date may contain Java byte code requiring at least a JVM 8.</p> <p>We therefore strongly recommend that you perform fresh installations only with Java 8, and update existing installations running with Java 7, to Java 8 as of now.</p> <p>See SAP Cloud Connector – Java 7 support will phase out and Update the Java VM.</p>	Announcement	2019-09-13

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.1 - Features	<p>Release of Cloud Connector version 2.12.1 introduces the following features and enhancements:</p> <ul style="list-style-type: none"> Subject Alternative Names are separated from the subject definition and provide enhanced configuration options. You can configure complex values easily when creating a certificate signing request. See Exchange UI Certificates in the Administration UI. In a high availability setup, the master instance detection no longer switches automatically if the configuration between the two instances is inconsistent. Disaster recovery switch back to main subaccount is periodically checked (if not successful) every 6 hours. Communication to on-premise systems supports SNI (Server Name Indication). 	New	2019-08-15
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.1 - Fixes	<p>Release of Cloud Connector version 2.12.1 provides the following bug fixes:</p> <ul style="list-style-type: none"> The communication between master and shadow instance no longer ends up in unusable clients that show 403 results due to CSRF (Cross-Site Request Forgery) failures, which could cause undesired role switches. When restoring a backup, the administrator password check works with all LDAP servers. The LDAP configuration test utility properly supports secure communication. The Refresh Subaccount Certificate dialog is no longer hanging when the refresh action fails due to some authentication or authorization issue. 	Changed	2019-08-15
Connectivity	Integration Suite	Cloud Foundry	Destination Service Scope Attribute for OAuth-based Authentication Types	<p>You can use the <code>scope destination</code> attribute for the OAuth-based authentication types <code>OAuth2ClientCredentials</code>, <code>OAuth2UserTokenExchange</code> and <code>OAuth2SAMLBearerAssertion</code>. This additional attribute provides flexibility on destination configuration level, letting you specify what scopes are selected when the OAuth access token is automatically retrieved by the service. See HTTP Destinations.</p>	New	2019-08-15

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo	JCo Runtime for SAP Cloud Platform - Features	<ul style="list-style-type: none"> Additional APIs have been added to <code>JCoBackgroundUnitAttributes</code>. See API documentation for details. If a structure or table contains only char-like fields, new APIs let you read or modify all of them at once for the structure or the current table row. See API documentation of <code>JCoTable</code> and <code>JCoStructure</code>. 	New	2019-07-18
Connectivity	Integration Suite	Neo	JCo Runtime for SAP Cloud Platform - Fixes	<ul style="list-style-type: none"> qRFC and tRFC requests sent to an ABAP system by JCo can be monitored again by AIF. Structure fields of type STRING are no longer truncated if there is a white space at the end of the field. 	Changed	2019-07-18
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - JCo Multitenancy	<p>The Connectivity service supports multitenancy for JCo applications.</p> <p>This feature requires a runtime environment with SAP Java Buildpack version 1.9.0 or higher.</p> <p>See Scenario: Multitenancy for JCo Applications (Advanced).</p>	New	2019-06-20
Connectivity	Integration Suite	Cloud Foundry	Cloud Cockpit - Cloud Connector View	The Cloud Connector view is available also for Cloud Foundry regions. It lets you see which Cloud Connectors are connected to a subaccount.	New	2019-04-25

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12 - Features	<p>Release of Cloud Connector version 2.12 introduces the following features and enhancements:</p> <ul style="list-style-type: none"> The administration UI is now accessible not only with an administrator role, but also with a display and a support role. See Configure Named Cloud Connector Users and Use LDAP for Authentication. For HTTP access control entries, you can <ul style="list-style-type: none"> allow a protocol upgrade, e.g. to WebSockets, for exposed resources. See Limit the Accessible Services for HTTP(S). define which host (virtual or internal) is sent in the host header. See Expose Intranet Systems, Step 8. A disaster recovery subaccount in disaster recovery mode can be converted into a standard subaccount, if a disaster recovery region replaces the original region permanently. See Convert a Disaster Recovery Subaccount into a Standard Subaccount. A service channel overview lets you check at a glance, which server ports are used by a Cloud Connector installation. See Service Channels: Port Overview. Important subaccount configuration can be exported, and imported into another subaccount. See Copy a Subaccount Configuration. An LDAP authentication configuration check lets you analyze and fix configuration issues before activating the LDAP authentication. See Use LDAP for Authentication. You can use different user roles to access the Cloud Connector configuration REST APIs. See Configuration REST APIs. REST APIs for shadow instance configuration have been added. See Shadow Instance Configuration. You can define scenarios for resources. Such a scenario can be exported, and imported into other hosts. See Configure Accessible Resources. 	New	2019-04-25

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12 - Fixes	<p>Release of Cloud Connector version 2.12 provides the following bug fixes:</p> <ul style="list-style-type: none"> The SAN (subjectAlternativeName) usage in certificates can be defined in a better way and is stored correctly in the certificate. See Exchange UI Certificates in the Administration UI. <code>IllegalArgumentExcept ion</code> does not occur anymore in HTTP processing, if the backend closes a connection and data are streamed. DNS caching is now recognized in reconnect situations if the IP of a DNS entry has changed. SNC with load balancing now works correctly for RFC SNC-based access control entries. A master-master situation is also recognized if, at startup of the former master instance, the new master (the former shadow instance) is not reachable. Solution management model generation works correctly for a shadow instance. The daemon is started properly on SLES 12 standard installations at system startup. 	Changed	2019-04-25
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Authentication Types	<p>Authentication type <code>OAuth2SAMLBearerAssertion</code> provides two different types of <code>Token Service URL</code>:</p> <ul style="list-style-type: none"> Dedicated: used in the context of a single tenant, or Common: used in the context of multiple tenants. <p>For type <code>Common</code>, the tenant subdomain is automatically set to the target <code>Token Service URL</code>.</p> <p>In addition, cloud applications can use the <code>x-user-token</code> HTTP header to propagate the user access token to the external target service at runtime. By default, the user principal is processed via the authorization HTTP header.</p> <p>See SAML Bearer Assertion Authentication.</p>	New	2019-04-11
Connectivity	Integration Suite	Neo Cloud Foundry	Connectivity Service - Fix	<p>When an on-premise system closed a connection that uses an RFC or SOCKS5 proxy, the Connectivity service kept the connection to the cloud application alive.</p> <p>This issue has been fixed. The connection is now always closed right after sending the response.</p>	Changed	2019-04-11

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Protocols	<p>The Connectivity service supports TCP connections to on-premise systems, exposing a SOCKS5 proxy to cloud applications.</p> <p>This feature follows the concept of binding the credentials of a Connectivity service instance.</p> <p>See Using the TCP Protocol for Cloud Applications.</p>	New	2019-03-14
Connectivity	Integration Suite	Neo	Connectivity Service - Fix	<p>After receiving an on-premise system response with HTTP header <i>Connection: close</i>, the Connectivity service kept the HTTP connection to the cloud application alive.</p> <p>This issue has been fixed. The connection is now always closed right after sending the response.</p>	Changed	2019-03-14
Connectivity	Integration Suite	Neo	Cloud Connector - Certificate Update	<p>For the Connectivity service (Neo environment), a new, region-specific certificate authority (X.509 certificate) is being introduced.</p> <p>If you use the Cloud Connector for on-premise connections to the Neo environment, you must import the new certificate authority into your trust configuration.</p> <ul style="list-style-type: none"> After the next month (concrete notification will be rolled out), the current certificate authority will no longer be used to issue client certificates for Cloud Connector deployments, and only the new one will be used. The Connectivity service will still trust client certificates of Cloud Connector deployments that were already issued. After a three-month period (concrete notification will be rolled out), that trust will be removed and your Cloud Connector deployment must be configured to use the new client certificates. <p>See Update the Certificate for a Subaccount.</p>	Announcement	2019-02-28
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Authentication Types	<p>The new authentication type <code>OAuth2UserTokenExchange</code> lets your applications use an automated exchange of user access tokens when accessing other applications or services. The feature supports single-tenant and multi-tenant scenarios. See OAuth User Token Exchange Authentication.</p>	New	2019-02-14
Connectivity	Integration Suite	Neo	RFC - Stateful Sequences	<p>You can make a stateful sequence of function module invocations work across several request/response cycles. See Invoking ABAP Function Modules via RFC.</p>	Changed	2019-01-31

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.11.3	A security note for Cloud Connector version 2.11.3 has been issued. See SAP note 2696233 .	Changed	2019-01-15
Connectivity	Integration Suite	Cloud Foundry	Protocols - RFC Communication	You can use the RFC protocol to set up communication with on-premise ABAP systems for applications in the Cloud Foundry environment. This feature requires a runtime environment with SAP Java Buildpack version 1.8.0 or higher. See Invoking ABAP Function Modules via RFC .	New	2019-01-17
Connectivity	Integration Suite	Cloud Foundry	Destinations - Renew Certificates	A button in the <i>Destinations</i> editor lets you update the validity period of an X.509 certificate. See Set up Trust Between Systems .	New	2019-01-17

1.1.2.4 2018 Connectivity (Archive)

2018

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo	Connectivity Service - Performance	A change in the SAP Cloud Platform Connectivity service improves performance of data upload (on-premise to cloud) and data download (cloud to on-premise) up to 4 times and 15-30 times respectively.	Changed	2018-12-20

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo	Connectivity Service - Resilience	The Connectivity service has a better protection against zombie connections, which improves resilience and overall availability for the cloud applications consuming it.	Changed	2018-12-20
Connectivity	Integration	Neo	Password Storage Service	A Password Storage REST API is available in the SAP API Business Hub, see Password Storage (Neo Environment) .	New	2018-12-06
Connectivity	Integration	Neo	Destination Configuration Service	A Destination Configuration service REST API is available in the SAP API Business Hub.	New	2018-12-06
Connectivity	Integration	Cloud Foundry	Destination Service	A Destination service REST API is available in the SAP API Business Hub.	New	2018-12-06
Connectivity	Integration	Neo	JCo Runtime for SAP Cloud Platform - Fixes	<ul style="list-style-type: none"> When using <code>JCoRecord.fromJSON()</code> for a structure parameter, the data is now always sent to the backend system. Also, you do not need to append the number of provided rows for table parameters before parsing the JSON document anymore. Depending on the configuration of certain JCo properties, an internally managed connection pool could throw a <code>JCoException</code> (error group <code>JCO_ERROR_RESOURCE</code>). In a thread waiting for a free connection from this pool, an error message then erroneously reported that the pool was exhausted. This error situation could occur if the used destination was not configured with the property <code>jco.destination.max_get_client_time</code> set to 0 and the destination's <code>jco.destination.peak_limit</code> value was set higher than the <code>jco.destination.pool_capacity</code>. This issue has been fixed. 	Changed	2018-12-06

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo	JCo Runtime for SAP Cloud Platform - Features	Support of the RFC fast serialization. Depending on the exchanged parameter and data types, the performance improvements for RFC communication can reach multiple factors. See SAP note 2372888 (prerequisites) and Parameters Influencing Communication Behavior (JCo configuration in SAP Cloud Platform).	Changed	2018-12-06
Connectivity	Integration	Neo	JCo Runtime for SAP Cloud Platform - Information	Local runtimes on Windows must install the VS 2013 redistributables for x64, instead of VS 2010.	Changed	2018-12-06
Connectivity	Integration	Neo Cloud Foundry	Cloud Connector Fixes	Release of Cloud Connector 2.11.3: <ul style="list-style-type: none"> An issue in RFC communication could cause the trace entry <i>com.sap.scc.jni.CpicCommunicationException: no SAP ErrInfo available</i> when the network is slow. This issue has been fixed. The Windows service no longer runs in <i>error 1067</i> when stopped by an administrator. In previous releases, the connection between a shadow and a master instance occasionally failed at startup and produced an empty error message. This issue has been fixed. The Cloud Connector does not cache Kerberos tokens in the protocol handler any more, as they are one-time tokens and cannot be reused. For HTTP access control entries, you can configure resources containing a # character. 	Changed	2018-12-06

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo Cloud Foundry	Cloud Connector Enhancements	<p>Release of Cloud Connector 2.11.3:</p> <ul style="list-style-type: none"> If the user <code>sapadm</code> exists on a system, the installation on Linux assigns it to the <code>sccgroup</code>, which is a prerequisite for solution management integration to work properly, see Configure Solution Management Integration [page 619]. Restoring a backup has been improved. See Configuration Backup [page 623]. The HTTP session store size has been reduced. You can handle higher loads with a given heap size. Cipher suite configuration has been improved. Also, there is a new security status entry for cipher suites, see Recommendations for Secure Setup [page 383]. 	Changed	2018-12-06
Connectivity	Integration	Neo	HTTP Destinations	<p>The <i>OAuth2 Client Credentials</i> grant type is supported by the <i>Destinations</i> editor in the SAP Cloud Platform cockpit as well as by the client Java APIs <code>ConnectivityConfiguration</code>, <code>AuthenticationHeaderProvider</code> and <code>HttpDestination</code>, available in SAP Cloud Platform Neo runtimes.</p> <p>See OAuth Client Credentials Authentication.</p>	Changed	2018-10-11
Connectivity	Integration	Cloud Foundry	User Propagation	<p>The connectivity service supports the SaaS application subscription flow and can be declared as a dependency in the get dependencies subscription callback, also via MTA (multi-target)-bundled applications.</p> <p>See Consuming the Connectivity Service (Cloud Foundry Environment) and Configure Principal Propagation via User Exchange Token (Cloud Foundry Environment).</p>	Changed	2018-09-27
Connectivity	Integration	Neo Cloud Foundry	Cloud Connector 2.11.2	<p>Release of Cloud Connector 2.11.2</p> <ul style="list-style-type: none"> SNC configuration now provides the value of the environment variable <code>SECUDIR</code>, which you need for the usage of the SAP Cryptographic Library (SAPCRYPTOLIB). See Initial Configuration (RFC). On Linux, the RPM (Red Hat Package Manager) now ensures that the configuration of the interaction with the SAP Host Agent (used for the Solution Manager integration) is adjusted. See Configure Solution Management Integration. The Cloud Connector shadow instance now provides a configuration option for the connection and request timeout that may occur during health check against the master instance. See Master and Shadow Administration. 	Changed	2018-08-16

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo Cloud Foundry	Cloud Connector 2.11.2	<p>Fixes of Cloud Connector 2.11.2</p> <ul style="list-style-type: none"> In a high availability setup, the switch from the master instance to the shadow instance occasionally caused communication errors towards on-premise systems. This issue has now been fixed. You can now import multiple certificates with the same subject to the trust store. Details about expiration date and issuer are displayed in the tool tip. See Set Up Trust, section <i>Trust Store</i>. You can now configure also the MOC (Multiple Origin Composition) OData service paths as resources. The <code>Location</code> header is now adjusted correctly according to your access control settings in case of a redirect. Principal propagation now also works with SAML assertions that contain an empty attribute element. SAP Cloud Platform applications occasionally got an HTTP 500 (internal server error) response when an HTTP connection was closed. The applications are now always informed properly. 	Changed	2018-08-16
Connectivity	Integration	Neo	HttpDestination Library	<p>The SAP <code>HttpDestination</code> library (available in the SDK and cloud runtime "Java EE 6 Web Profile") now creates Apache <code>HttpClient</code> instances which work with strict SNI (Server Name Indication) servers.</p> <p>Use cases with strict SNI configuration on the server side will no longer get the error message <i>Failure reason: "peer not authenticated"</i>, that was raised either at runtime or while performing a connection test via the SAP Cloud Platform cockpit Destinations editor (Check Connection function).</p>	Changed	2018-08-16

1.1.2.5 2017 Connectivity (Archive)

Archived release notes for 2017 and older.

28 September 2017 - Connectivity

New

The destination service (Beta) is available in the Cloud Foundry environment. See [Consuming the Destination Service \[page 243\]](#).

3 August 2017 - Connectivity

Enhancement

Cloud Connector

Release of SAP Cloud Platform Cloud Connector 2.10.1.

- The URLs of HTTP requests can now be longer than 4096 bytes.
 - SAP Solution Manager can be integrated with one click of a button if the host agent is installed on a Cloud Connector machine. See the *Solution Management* section in [Monitoring \[page 660\]](#).
 - The limitation that only 100 subaccounts could be managed with the administration UI has been removed. See [Managing Subaccounts \[page 401\]](#).
-

Fix

Cloud Connector

- The regression of 2.10.0 has been fixed, as principal propagation now works for RFC.
 - The cloud user store works with group names that contain a backslash (\) or a slash (/).
 - Proxy challenges for NT LAN Manager (NTLM) authentication are ignored in favor of Basic authentication.
 - The back-end connection monitor works when using a JVM 7 as a runtime of Cloud Connector.
-

25 May 2017 - Connectivity

Enhancement

Cloud Connector

Release of SAP HANA Cloud connector 2.10.0.1

- Support of connectivity to an SAP Cloud Platform Cloud Foundry environment.
- Support of direct connectivity with S/4HANA Cloud systems. You can open a Service Channel to an S/4HANA Cloud system in order to use Communication Scenarios requiring RFC communication to S/4HANA Cloud. See [Configure a Service Channel for RFC \[page 611\]](#).
- Support of arbitrary protocols via the possibility to configure a TCP access control entry. SAP Cloud Platform Connectivity is offering a SOCKS5 proxy, with which you can address such exposed hosts. See [Using the TCP Protocol for Cloud Applications](#).
- Support for disaster recovery events of SAP Cloud Platform regions. For each subaccount you can configure a disaster recovery subaccount for a disaster region. In case of a disaster, the disaster recovery account can be switched active immediately using the exact same configuration. See [Configure a Disaster Recovery Subaccount \[page 415\]](#).
- In the access control settings you can add further constraints for RFC based communication to ABAP systems: an administrator can configure, which clients shall be exposed and can define which users should not be able to access the system via Cloud Connector. See [Configure Access Control \(RFC\) \[page 465\]](#).
- You can generate self-signed certificates for CA and system certificate so that you can setup demo scenarios with principal propagation without the need of using lengthy openssl or keytool command sequences. See [Configure a CA Certificate \[page 424\]](#) and [Initial Configuration \(HTTP\) \[page 396\]](#).
- A first set of monitoring HTTP APIs have been introduced: The state of all subaccount connections, a back-end connection monitor and a performance overview monitor. See [Monitoring APIs \[page 670\]](#).
- On Windows platforms, Cloud Connector 2.10 now requires Visual Studio 2013 runtime libraries.

Fix

Cloud Connector

- There is no longer a bottleneck that could lengthen the processing times of requests to exposed back-end systems, after many hours under high load when using principal propagation, connection pooling, and many concurrent sessions.
- Session management is no longer terminating early active sessions in principal propagation scenarios.
- On Windows 10 hardware metering in virtualized environments shows hard disk and CPU data.

11 May 2017 - Connectivity

New

In case the remote server supports only TLS 1.2, use this property to ensure that your scenario will work. As TLS 1.2 is more secure than TLS 1.1, the default version used by HTTP destinations, consider switching to TLS 1.2.

30 March 2017 - Connectivity

Enhancement

The release of SAP Cloud Platform Cloud Connector 2.9.1 includes the following improvements:

- UI renovations based on collected customer feedback. The changes include rounding offs, fixes of wrong/odd behaviors, and adjustments of controls. For example, in some places tables were replaced by *sap.ui.table.Table* for better experience with many entries.
 - You can trigger the creation of a thread dump from the *Log and Trace Files* view.
 - The connection monitor graphic for idle connections was made easier to understand.
-

Fix

- When configuring authentication for LDAP, the alternate host settings are no longer ignored.
 - The email configuration for alerts is processing correctly the user and password for access to the email server.
 - Some servers used to fail to process HTTP requests when using the HTTP proxy approach ([HTTP Proxy for On-Premise Connectivity](#)) on the SAP Cloud Platform side.
 - A bottleneck was removed that could lengthen the processing times of requests to exposed back-end systems under high load when using principal propagation.
 - The Cloud Connector accepts passwords that contain the '\$' character when using authentication-mode password.
-

16 March 2017 - Connectivity

Enhancement

Update of JCo runtime for SAP Cloud Platform. See [Connectivity \[page 4\]](#).

Fix

A `java.lang.NullPointerException` might have occurred when using a `JCoRepository` instance in roundtrip optimization mode (will be used if the JCo property `jco.use_repository_roundtrip_optimization` was set to 1 at its creation time). The `NullPointerException` was either thrown when trying to execute a `JCoFunction` object, which has been created by such a repository instance, or even earlier when querying the meta data for a `JCoFunction`, `JCoFunctionTemplate` or a `JCoRecordMetaData` object from an AS ABAP back-end system. Only certain complex data structures and table parameter definitions were affected by this bug.

Older Release Notes

- [2016](#) 
- [2015](#) 
- [2014](#) 
- [2013](#) 

1.1.3 Administration

Manage destinations and authentication for applications in the Cloud Foundry environment.

Task	Description
Managing Destinations [page 59]	Manage HTTP destinations for Cloud Foundry applications in the SAP BTP cockpit.
HTTP Destinations [page 86]	You can choose from a broad range of authentication types for HTTP destinations, to meet the requirements of your specific communication scenario.
RFC Destinations [page 157]	Use RFC destinations to communicate with an on-premise ABAP system via the RFC protocol.
Principal Propagation [page 168]	Use principal propagation (user propagation) to securely forward cloud users to a back-end system (single sign-on).
Set up Trust Between Systems [page 172]	Download and configure X.509 certificates as a prerequisite for user propagation from the Cloud Foundry environment to the Neo environment or to a remote system outside SAP BTP, like S/4HANA Cloud, C4C, Success Factors, and others.
Multitenancy in the Connectivity Service [page 203]	Manage destinations for multitenancy-enabled applications that require a connection to a remote service or on-premise application.
Create and Bind a Connectivity Service Instance [page 206]	To use the Connectivity service in your application, you must first create and bind an instance of the service.

Task	Description
Create and Bind a Destination Service Instance [page 209]	To use the Destination service in your application, you must first create and bind an instance of the service.
Configuring Backup [page 213]	Create a backup of your destination configurations.
Destination Fragments [page 213]	Use destination fragments to override and/or extend destination properties as part of the "Find Destination" call.

1.1.3.1 Managing Destinations

To manage destinations for your application, choose a procedure that fits best your requirements.

There are various ways to manage destinations. Each method is characterized by different prerequisites and limitations. Before choosing a method, you should evaluate them and decide which one is the most appropriate for your particular scenario. The following table compares the available methods:

Method	Create from Scratch	Create from Template	Create from File (Import)	Save to File (Export)	Update	Delete	Clone	Check Connection
Using the Destinations Editor in the Cockpit [page 60]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Destination Service REST API [page 80]	Yes	No	Yes	Yes	Yes	Yes	No	No
Create Destinations Using the MTA Descriptor [page 81]	Yes	Limited	No	No	Yes	No	No	No
Create Destinations on Service Instance Creation [page 86]	Yes	No	No	No	Yes	No	No	No

1.1.3.1.1 Using the Destinations Editor in the Cockpit

Use the [Destinations](#) editor in the SAP BTP cockpit to configure HTTP, RFC or mail destinations in the Cloud Foundry environment.

The [Destinations](#) editor lets you manage destinations on subaccount or service instance level.

You can use a destination to:

- Connect your Cloud Foundry application to the **Internet** (via HTTP), as well as to an **on-premise system** (via HTTP or RFC).
- Send and retrieve e-mails, configuring a mail destination.
- Create a destination for subscription-based scenarios, pointing to your service instance. For more information, see [Destinations Pointing to Service Instances \[page 71\]](#).

Prerequisites

1. You are logged into the SAP BTP cockpit.
2. You have the required authorizations. See [User Roles \[page 11\]](#).
3. Make sure the following is fulfilled:
 - Service instance level – you must have created a Destination service instance, see [Create and Bind a Destination Service Instance \[page 209\]](#).
 - Subaccount level – no specific prerequisites.

For more information, see [Access the Destinations Editor \[page 61\]](#).

Restrictions

- A destination name must be unique for the current application. It must contain only alphanumeric characters, underscores, and dashes. The maximum length is 200 characters.
- The currently supported destination types are **HTTP**, **RFC** and **MAIL**.
 - [HTTP Destinations \[page 86\]](#) - provide data communication via the HTTP protocol and are used for both Internet and on-premise connections.
 - [RFC Destinations \[page 157\]](#) - make connections to ABAP on-premise systems via RFC protocol using the Java Connector (JCo) as API.
 - Mail destinations - specify an e-mail provider for sending and retrieving e-mails.

Tasks

- [Create Destinations from Scratch \[page 63\]](#)
- [Create Destinations from a Template \[page 71\]](#)

- [Check the Availability of a Destination \[page 73\]](#)
- [Clone Destinations \[page 74\]](#)
- [Edit and Delete Destinations \[page 75\]](#)
- [Use Destination Certificates \[page 76\]](#)
- [Import Destinations \[page 79\]](#)
- [Export Destinations \[page 80\]](#)

Related Information

[Destination Examples \[page 68\]](#)

1.1.3.1.1.1 Access the Destinations Editor

Access the Destinations Editor in the SAP BTP cockpit to create and manage destinations in the Cloud Foundry environment.

You can edit destinations at two different levels:

- Subaccount level
- Service instance level

On subaccount level, you can specify a destination for the entire subaccount, defining the used communication protocol and more properties, like authentication method, proxy type and URL.

On service instance level, you can reuse this destination for a specific space and adjust the URL if required. You can also create a new destination only on service instance level that is specific to the selected service instance and its assigned applications.

Prerequisites

- You are logged into the SAP BTP cockpit.
- You have the required authorizations. See [User Roles \[page 11\]](#).

Procedure

Access on Subaccount Level

1. In the cockpit, select your *Global Account* and your subaccount name from the *Subaccount* menu in the breadcrumbs.

- From the left-side panel, choose **Connectivity > Destinations**.

Access on Service Instance Level

Note

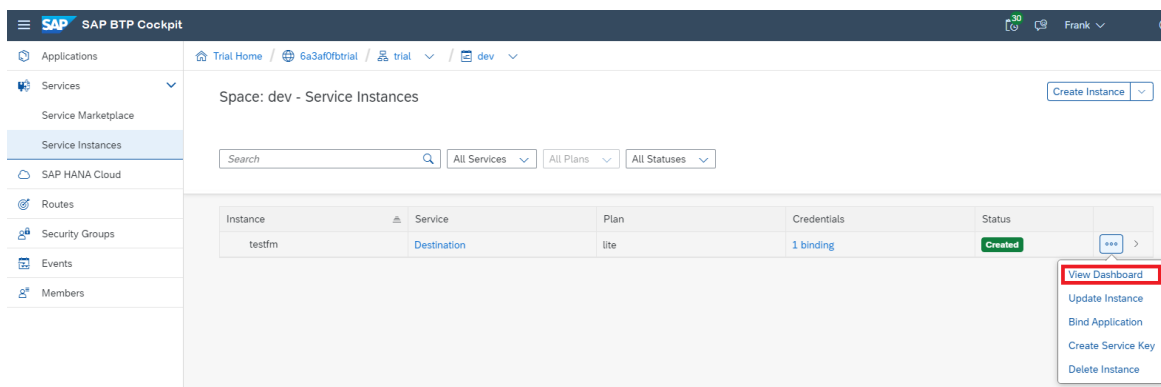
To perform these steps, you must have created a Destination service instance in your space, see [Create and Bind a Destination Service Instance \[page 209\]](#). On service instance level, you can set destinations only for Destination service instances.

- In the cockpit, choose your *Global Account* from the *Region Overview* and select a *Subaccount*.

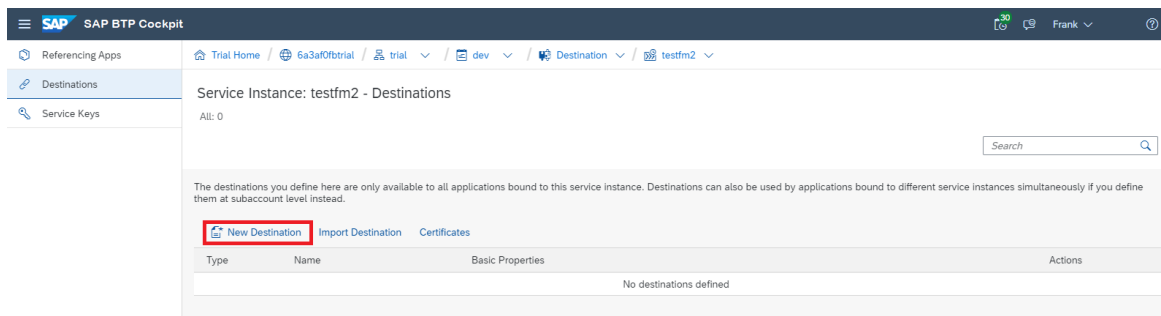
Note

The Cloud Foundry Organization must be enabled.

- From the *Spaces* section, select a space name.
- From the left-side menu, choose **Services > Service Instances**.
- Choose the *Actions* icon for a Destination service instance and select *View Dashboard*.



- On the *Destinations* screen, you can create new destinations or edit existing ones.



See also section *Create and Bind a Service Instance from the Cockpit* in [Create and Bind a Destination Service Instance \[page 209\]](#).

Related Information

[Create Destinations from Scratch \[page 63\]](#)

[Create Destinations from a Template \[page 71\]](#)
[Check the Availability of a Destination \[page 73\]](#)
[Clone Destinations \[page 74\]](#)
[Edit and Delete Destinations \[page 75\]](#)
[Use Destination Certificates \[page 76\]](#)
[Import Destinations \[page 79\]](#)
[Export Destinations \[page 80\]](#)

1.1.3.1.1.2 Create Destinations from Scratch

Use the *Destinations* editor in the SAP BTP cockpit to configure destinations from scratch.

Configuring destinations from scratch provides the complete set of editing functions. While this requires deeper technical knowledge of the scenario and the required connection configuration, it is the most flexible procedure and lets you create any type of supported destination.

To [Create Destinations from a Template \[page 71\]](#), in contrast, you do not need this deeper knowledge for configuration, but editing options are limited by the available templates.

Related Information

[Create HTTP Destinations \[page 63\]](#)
[Create RFC Destinations \[page 65\]](#)
[Create Mail Destinations \[page 67\]](#)
[Destination Examples \[page 68\]](#)

1.1.3.1.1.2.1 Create HTTP Destinations

Create HTTP destinations in the *Destinations* editor (SAP BTP cockpit).

Prerequisites

You have logged into the cockpit and opened the *Destinations* editor.

Procedure

1. Choose [New Destination](#).

Note

In section **Destination Configuration**, do not change the default tab [Blank Template](#), unless you want to create a destination for a specific service instance in a subscription-based scenario. For more information, see [Destinations Pointing to Service Instances \[page 71\]](#).

2. Enter a destination name.
3. From the [<Type>](#) dropdown menu, choose **HTTP**.
4. The [<Description>](#) field is optional.
5. Specify the destination URL.
6. From the [<Proxy Type>](#) dropdown box, select **Internet** or **OnPremise**, depending on the connection you need to provide for your application.


Note

For more information, see also [HTTP Destinations \[page 86\]](#).

7. From the [<Authentication>](#) dropdown box, select the authentication type you need for the connection. For details, see [HTTP Destinations \[page 86\]](#).

Note

If you set an **HTTPS** destination, you need to also add a Trust Store. For more information, see [Use Destination Certificates \[page 76\]](#).

8. (Optional) If you are using more than one Cloud Connector for your subaccount, you must enter the [<Location ID>](#) of the target Cloud Connector. See also [Managing Subaccounts \[page 401\]](#) (section **Procedure**, step 4).
9. (Optional) You can enter additional properties.
 - a. In the [Additional Properties](#) panel, choose [New Property](#).
 - b. Enter a key (name) or choose one from the dropdown menu and specify a value for the property. You can add as many properties as you need.
 - c. To delete a property, choose the  button next to it.

Note

For a detailed description of specific properties for SAP Business Application Studio (formerly known as SAP Web IDE), see [Connecting to External Systems](#).

10. When you are ready, choose the [Save](#) button.

Related Information

[Edit and Delete Destinations \[page 75\]](#)

1.1.3.1.1.2.2 Create RFC Destinations

How to create RFC destinations in the *Destinations* editor (SAP BTP cockpit).

Prerequisites

You have logged into the cockpit and opened the *Destinations* editor.

Procedure

1. Choose *New Destination*.

Note

In section **Destination Configuration**, do not change the default tab *Blank Template*. Tab *Service Instance* only applies for HTTP destinations.

2. Enter a destination name.
3. From the *<Type>* dropdown menu, choose **RFC**.
4. The *<Description>* field is optional.
5. From the *<Proxy Type>* dropdown box, select **Internet** or **OnPremise**, depending on the connection you need to provide for your application.

Note

Using *<Proxy Type>* **Internet**, you can connect your application to any target service that is exposed to the Internet. *<Proxy Type>* **OnPremise** requires the Cloud Connector to access resources within your on-premise network.

6. Enter credentials for *<User>* and *<Password>*.
7. (Optional) Enter an *<Alias User>* name. See also [User Logon Properties \[page 158\]](#).
8. (Optional) Enter credentials for *<Repository User>* and *<Repository Password>*, if you want to use a different user for repository lookups. See also [Repository Configuration \[page 162\]](#).
9. (Optional) If you are using more than one Cloud Connector for your subaccount, you must enter the *<Location ID>* of the target Cloud Connector.
See also [Managing Subaccounts \[page 401\]](#) (section **Procedure**, step 4).
10. Depending on the proxy type of your RFC destination, specify at least the following JCo properties in section *Additional Properties*.
 - a. In the *Additional Properties* panel, choose *New Property*.
 - b. Add each required property from the dropdown menu and specify its value:

Proxy Type	Property	Description
OnPremise	Load Balancing Connections	
	<code>jco.client.r3name</code>	Three-letter system ID of your back-end ABAP system (as configured in the Cloud Connector).
	<code>jco.client.mshost</code>	Message server host (as configured in the Cloud Connector).
	<code>jco.client.group</code>	(Optional) The group of application servers that is used (logon group). If not specified, the group PUBLIC is used.
	<code>jco.client.client</code>	Three-digit ABAP client number (defines the client of the target ABAP system).
	Direct Connections	
	<code>jco.client.ashost</code>	Application server name of your target ABAP system (as configured in the Cloud Connector).
	<code>jco.client.sysnr</code>	Instance number of the application server (as configured in the Cloud Connector).
	<code>jco.client.client</code>	Three-digit ABAP client number (defines the client of the target ABAP system).
	Internet	<code>jco.client.wshost</code>
<code>jco.client.wsport</code>		WebSocket RFC server port on which the target ABAP system is listening.
<code>jco.client.client</code>		Three-digit ABAP client number (defines the client of the target ABAP system).

For a detailed description of RFC-specific properties (JCo properties), see [RFC Destinations \[page 157\]](#).

11. Press [Save](#).

Related Information

[Edit and Delete Destinations \[page 75\]](#)

[Destination Examples \[page 68\]](#)

[Cloud Connector \[page 343\]](#)

1.1.3.1.1.2.3 Create Mail Destinations

Create mail destinations in the [Destinations](#) editor (SAP BTP cockpit).

Prerequisites

You have logged into the cockpit and opened the [Destinations](#) editor.

Procedure

1. Choose [New Destination](#).

Note

In section **Destination Configuration**, do not change the default tab [Blank Template](#). Tab [Service Instance](#) only applies for HTTP destinations.

2. Enter a destination name.
3. From the [Type](#) dropdown menu, choose **MAIL**.
4. The [Description](#) field is optional.
5. From the [Proxy Type](#) dropdown box, select **Internet** or **OnPremise**, depending on the connection you need to provide for your application.

Note

To access a mail server located in your own network (via Cloud Connector), choose **OnPremise**. To access an external mail server, choose **Internet**.

6. Choose the [Authentication](#) type to be used for the destination and enter the required parameters. For a detailed parameter description, see [Configuring Authentication \[page 87\]](#).
7. Enter the additional property `mail.smtp.host` to specify the address of the target mail server.
 - a. In the [Additional Properties](#) panel, choose [New Property](#).
 - b. Choose `mail.smtp.host` from the dropdown menu and specify a value for the property.
8. (Optional) You can enter more additional properties.
 - a. In the [Additional Properties](#) panel, choose [New Property](#).
 - b. Enter a key (name) or choose one from the dropdown menu and specify a value for the property. You can add as many properties as you need. Each key of an additional property must start with "mail.".
 - c. To remove a property, choose the [Remove](#) button next to it.
9. When you are done, choose [Save](#).

Related Information

[Edit and Delete Destinations \[page 75\]](#)

[Destination Examples \[page 68\]](#)

[Cloud Connector \[page 343\]](#)

1.1.3.1.1.2.4 Destination Examples

Find configuration examples for HTTP and RFC destinations in the Cloud Foundry environment, using different authentication types.

Content

[HTTP Destination \(Internet, Client Certificate Authentication\) \[page 68\]](#)

[HTTP Destination \(Internet, OAuth2SAMLBearerAssertion\) \[page 69\]](#)

[HTTP Destination \(On-Premise\) \[page 69\]](#)

[RFC Destination \[page 70\]](#)

[Mail Destination \(Internet\) \[page 70\]](#)

[Mail Destination \(On-Premise\) \[page 70\]](#)

Example: HTTP Destination (Internet, Client Certificate Authentication)

The screenshot shows the SAP BTP Destination Configuration interface. At the top, there are navigation tabs: "New Destination", "Import Destination", "Certificates", "Download Trust", and "Renew Trust". Below this is a table with columns "Type", "Name", "Basic Properties", and "Actions". The table is currently empty, with the text "No destinations defined" centered below it. The main section is titled "Destination Configuration" and contains several input fields and dropdown menus. On the left, there are fields for "Name" (MyInternet), "Type" (HTTP), "Description", "URL" (https://google.com), "Proxy Type" (Internet), "Authentication" (ClientCertificateAuthentication), "Key Store Location" (MyKeyStore.jks), and "Key Store Password" (masked with asterisks). On the right, there are "Additional Properties" and "Upload and Delete Certificates" sections. The "Additional Properties" section includes a "Trust Store Location" dropdown (MyTrustStore.jks) and a "Trust Store Password" field. The "Upload and Delete Certificates" section has a "New Property" button and a "Use default JDK truststore" checkbox. At the bottom left, there are "Save" and "Cancel" buttons.

[Back to Content \[page 68\]](#)

Example: HTTP Destination (Internet, OAuth2SAMLBearerAssertion)

Destination Configuration

Name: MyDestination
Type: HTTP
Description: test
URL: https://sample.server.com
Proxy Type: Internet
Authentication: OAuth2SAMLBearerAssertion
Key Store Location:
Key Store Password:
Audience: https://myapp.sample.server.com
Client Key: *****
Token Service URL: https://sample.oauth.server.com
Token Service URL Type: Dedicated Common
Token Service User: <optional>
Token Service Password: <optional>
System User: <optional>

Additional Properties
 Use default JDK truststore

Upload and Delete Certificates

Save Cancel

New Property

[Back to Content \[page 68\]](#)

Example: HTTP Destination (On-Premise)

New Destination Import Destination Certificates Download Trust Renew Trust

Type	Name	Basic Properties	Actions
		No destinations defined	

Destination Configuration

Name: MyOnPremiseDestination
Type: HTTP
Description: test
Location ID: <optional>
URL: https://mycompany.corp:443
Proxy Type: OnPremise
Authentication: BasicAuthentication
User: SomeUser
Password: *****

Additional Properties

Save Cancel

New Property

[Back to Content \[page 68\]](#)

Example: RFC Destination

The screenshot shows the 'Destination Configuration' window for an RFC destination. The main properties are:

- Name: MyJCoSystem
- Type: RFC
- Description: (empty)
- Location ID: (empty)
- User: MyJCoUser
- Password: (masked with asterisks)
- Repository User: (empty)
- Repository Password: (empty)

The Additional Properties section includes:

- jco.client.lang: EN
- jco.client.ashost: abapserver.hana.cloud
- jco.client.client: 000
- jco.client.sysnr: 42

The following main properties correspond to the relevant additional properties:

User → `jco.client.user`

Password → `jco.client.passwd`

Repository password → `jco.destination.repository.passwd`

Note

For security reasons, do not use these additional properties but use the corresponding main properties' fields.

Back to [Content \[page 68\]](#)

Example: Mail Destination (Internet)

The screenshot shows the 'Destination Configuration' window for a Mail Destination (Internet). The main properties are:

- Name: mail_internet
- Type: MAIL
- Description: (empty)
- Proxy Type: Internet
- User: user
- Password: (masked with asterisks)

The Additional Properties section includes:

- mail.smtp.host: my-mail-server.com
- mail.transpor...: smtp

Back to [Content \[page 68\]](#)

Example: Mail Destination (On-Premise)

Destination Configuration

Name:	mail_onprem	Additional Properties		New Property
Type:	MAIL	mail.smtp.host	my-mail-server.com	
Description:		mail.transpor...	smtp	
Proxy Type:	OnPremise			
User:	user			
Password:	*****			
Location ID:	Paris			

[Back to Content \[page 68\]](#)

Related Information

[HTTP Destinations \[page 86\]](#)

[RFC Destinations \[page 157\]](#)

1.1.3.1.1.3 Create Destinations from a Template

Use a template to configure destinations with scenario-specific input data in the SAP BTP cockpit.

If you want to create several destinations for a common scenario, you can use a template that provides the scenario-specific input data. The Destination service uses the template to configure the destinations accordingly.

Currently, the following template is available for destination configuration:

[Destinations Pointing to Service Instances \[page 71\]](#)

1.1.3.1.1.3.1 Destinations Pointing to Service Instances

Create a destination for subscription-based scenarios that points to your service instance.

Note

This feature is applicable for a selected set of the most commonly used services (from a Destination service perspective). If you would like to use this feature for a service which is not yet supported, let us know by opening a support ticket, see [Connectivity Support \[page 869\]](#).

In the meantime, you can follow the steps described in [Create Destinations from Scratch \[page 63\]](#).

Usually, in the Cloud Foundry environment, you consume service instances by binding them to your applications. However, in subscription-based scenarios this is not always possible. If you have purchased a subscription to an SaaS application that runs in a provider's subaccount, you cannot bind your service instance to this application.

In this case, you must create a destination that points to your service instance. Applications can consume this destination through a subscription to gain access to your service instance.

If you create such a destination from scratch, you must provide a service key for your instance, look up the credentials, and enter these values in the newly created destination.

Using the *Destinations Pointing to Service Instances* template, you only have to select the corresponding service instance.

Note

This procedure only applies for HTTP destinations on subaccount level.

Prerequisites

- You have a service instance which you want to make accessible to applications you are subscribed to.
- You have the Space Developer role in the space where this service instance resides.
- You have logged in to the cockpit and opened the *Destinations* editor on *subaccount* level. See [Access the Destinations Editor \[page 61\]](#).

Procedure

1. Choose *New Destination*.
2. Select the tab *Service Instance* in the **Destination Configuration** section.
3. In the *<Service Instance>* dropdown list, you find all the service instances, grouped by space, where you have the role Space Developer.
4. Select a service instance.
5. Give the destination configuration a name and, optionally, a description.
6. (Optional) You can specify additional properties.
7. Choose *Next*.
A service key for that service instance is automatically generated, using the naming convention *<service_instance_name>-service-key*. If the key name already exists, it is reused. A new destination with pre-filled fields is previewed, using the given service instance data. Do not change the values of these fields.
8. If you want to create a destination with these values, choose *Save*. Otherwise, choose *Cancel*.

Result

You have a destination pointing to your service instance. If you delete this service instance or its service key, the destination stops working.

⚠ Caution

If you delete this service instance or its service key, the destination will stop working.

1.1.3.1.1.4 Check the Availability of a Destination

How to check the availability of a destination in the *Destinations* editor (SAP BTP cockpit).

Prerequisites

You have logged into the cockpit and opened the *Destinations* editor.

Context

You can use the *Check Connection* button in the *Destinations* editor of the cockpit to verify if the **URL** configured for an HTTP Destination is reachable and if the connection to the specified system is possible.

📘 Note

This check is available with *Cloud Connector version 2.7.1 or higher*.

For each destination, the check button is available in the destination detail view and in the destination overview list (icon *Check availability of destination connection* in section *Actions*).

📘 Note

The check does not guarantee that the target system or service is operational. It only verifies if a connection is possible.

This check is supported for destinations with <Proxy Type> **Internet** and **OnPremise**:

- For **Internet** destinations:
 - If the check receives an **HTTP status code above or equal to 500** from the targeted URL, the check is considered **failed**.
 - Every **HTTP status code below 500** is treated as **successful**.

⚠ Restriction

The targeted URL cannot be a private endpoint (for example, *localhost*).

- For **OnPremise** destinations:

- If the targeted backend is reached and returns an **HTTP status code below 500** the check is considered **successful**.

Error Messages for OnPremise Destinations

Error Message	Reason	Action
<i>Backend status could not be determined.</i>	<ul style="list-style-type: none"> • The Cloud Connector version is less than 2.7.1. • The Cloud Connector is not connected to the subaccount. • The backend returns an HTTP status code above or equal to 500 (server error). • The Cloud Connector is not configured properly. 	<ul style="list-style-type: none"> • Upgrade the Cloud Connector to version 2.7.1 or higher. • Connect the Cloud Connector to the corresponding subaccount. • Check the server status (availability) of the backend system. • Check the basic Cloud Connector configuration steps: Initial Configuration [page 388]
<i>Backend is not available in the list of defined system mappings in Cloud Connector.</i>	The Cloud Connector is not configured properly.	Check the basic Cloud Connector configuration steps: Initial Configuration [page 388]
<i>Resource is not accessible in Cloud Connector or backend is not reachable.</i>	The Cloud Connector is not configured properly.	Check the basic Cloud Connector configuration steps: Initial Configuration [page 388]
<i>Backend is not reachable from Cloud Connector.</i>	Cloud connector configuration is ok but the backend is not reachable.	Check the backend (server) availability.


1.1.3.1.1.5 Clone Destinations

How to clone destinations in the [Destinations](#) editor (SAP BTP cockpit).

Prerequisites

You have previously created or imported an HTTP destination in the [Destinations](#) editor of the cockpit.

Procedure

1. In the *Destinations* editor, go to the existing destination which you want to clone.
2. Choose the  icon.
3. The editor automatically creates and opens a new destination that contains all the properties of the selected one.
4. You can modify some parameters if you need.
5. When you are ready, choose the *Save* button.

Related Information

[Export Destinations \[page 80\]](#)

[Destination Examples \[page 68\]](#)


1.1.3.1.1.6 Edit and Delete Destinations

How to edit and delete destinations in the *Destinations* editor (SAP BTP cockpit).

Prerequisites

You have previously created or imported an HTTP destination in the *Destinations* editor of the cockpit.

Procedure

- Edit a destination:
 1. To edit a created or imported destination, choose the  button.
 2. You can edit the main parameters as well as the additional properties of a destination.
 3. Choose the *Save* button. The changes will take effect in up to five minutes.

→ Tip

For complete consistency, we recommend that you first stop your application, then apply your destination changes, and then start again the application. Also, bear in mind that these steps will cause application downtime.

- Delete a destination:

To remove an existing destination, choose the  button. The changes will take effect in up to five minutes.

Related Information

[Export Destinations \[page 80\]](#)

[Destination Examples \[page 68\]](#)

1.1.3.1.1.7 Use Destination Certificates

Maintain trust store and key store certificates in the *Destinations* editor (SAP BTP cockpit).

Prerequisites

You have logged on to the cockpit and opened the *Destinations* editor. For more information, see [Access the Destinations Editor \[page 61\]](#).

Context

⚠ Caution

Uploaded certificates are accessible via the REST APIs, including any private data they may contain.

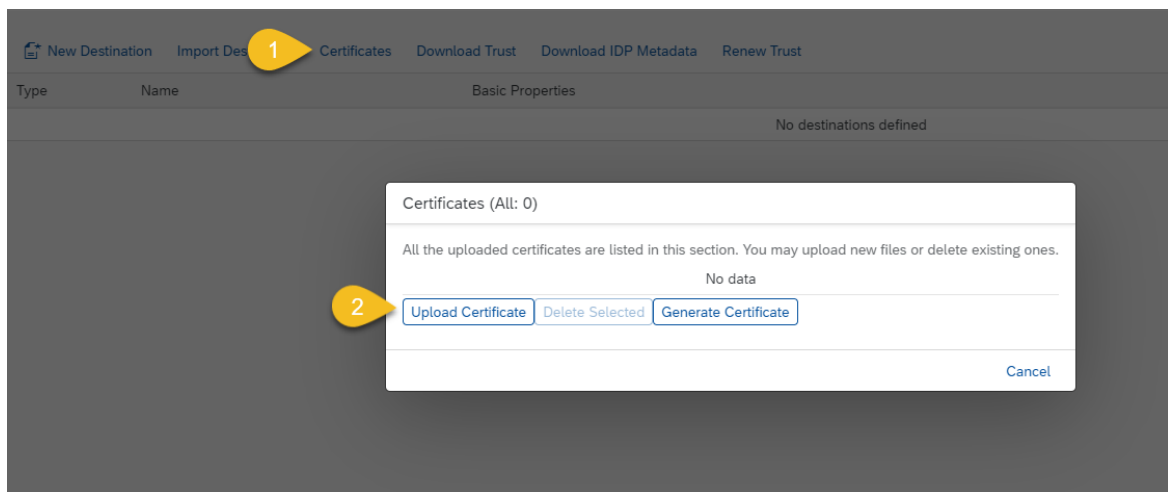
You can upload, add and delete certificates for your connectivity destinations.

- You can use JKS, PFX, PEM and P12 files.
- You can add certificates only for **HTTPS** destinations. The trust store can be used for all authentication types. A key store is available only for `ClientCertificateAuthentication` and OAuth authentication types (as a token service keystore, see [OAuth with X.509 Client Certificates \[page 133\]](#)).
- An uploaded certificate file should contain the entire certificate chain.

Procedure

Upload Certificates

1. Choose *Certificates*.
2. Choose *Upload Certificate*.



3. Browse to the certificate file you need to upload.
The certificate file is added.

Note

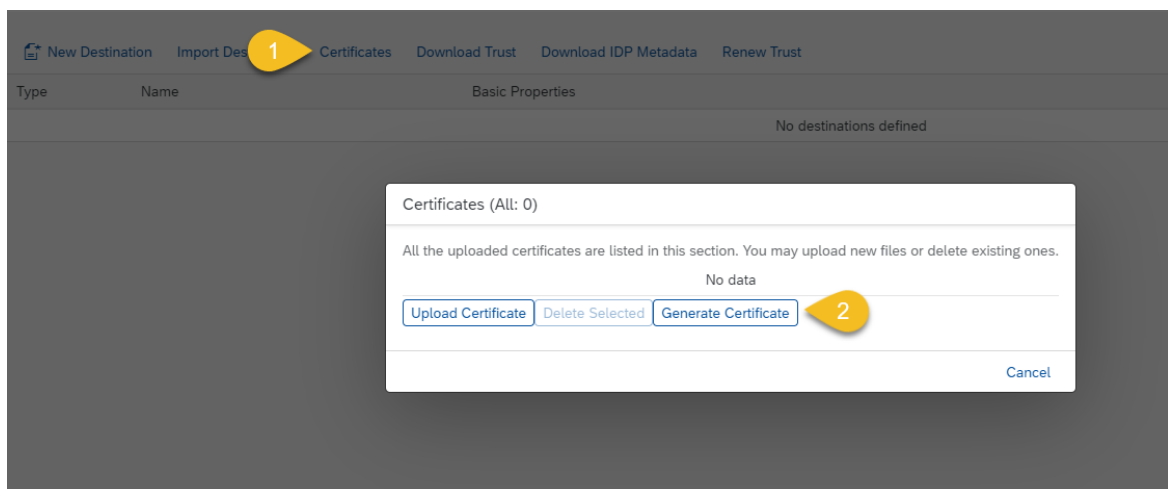
You can upload a certificate during creation or editing of a destination, by clicking the [Upload and Delete Certificates](#) link.

Caution

Certificates added through the [Upload Certificate](#) option cannot be renewed automatically.

Generate a SAP Cloud PKI Infrastructure Certificate

1. Choose [Certificates](#).
2. Choose [Generate Certificate](#).



3. In the pop-up window, enter certificate name and type. Optionally, you can enter certificate CN and certificate validity. (Optional) Additionally, you can select the [Enable automatic renewal](#) checkbox to renew

the certificate automatically when it nears its expiration date. Choose [Generate Certificate](#) again. The certificate is generated, and you can use it in your destinations.

Generate new certificate

Generated certificate will be part of SAP Cloud PKI infrastructure.

Certificate File Name *

File Name Extension * ▼

Certificate Common Name

Certificate Validity Time Unit ▼

Certificate Validity Value

Password

Enable automatic renewal

3

More Information

[Create HTTP Destinations \[page 63\]](#)

[Edit and Delete Destinations \[page 75\]](#)

[Import Destinations \[page 79\]](#)

[Set up Trust Between Systems \[page 172\]](#)

1.1.3.1.1.8 Import Destinations

How to import destinations in the *Destinations* editor (SAP BTP cockpit).

Prerequisites

You have previously created an HTTP destination.

Note

The *Destinations* editor allows importing destination files with extension `.props`, `.properties`, `.jks`, and `.txt`, as well as files with no extension. Destination files must be encoded in **ISO 8859-1** character encoding.

Procedure

1. Log into the cockpit and open the *Destinations* editor.
2. Choose *Import from File*.
3. Browse to a configuration file that contains destination configuration.
 - If the configuration file contains valid data, it is displayed in the *Destinations* editor with no errors. The *Save* button is enabled so that you can successfully save the imported destination.
 - If the configuration file contains invalid properties or values, under the relevant fields in the *Destinations* editor are displayed error messages in red which prompt you to correct them accordingly.

Related Information

[Edit and Delete Destinations \[page 75\]](#)

[Destination Examples \[page 68\]](#)


1.1.3.1.1.9 Export Destinations

Export destinations from the *Destinations* editor in the SAP BTP cockpit to backup or reuse a destination configuration.

Prerequisites

You have created a destination in the *Destinations* editor.

Procedure

1. Log into the cockpit and open the *Destinations* editor.
2. Select a destination and choose the  button.
3. Browse to the location on your local file system where you want to save the new destination.
 - If the destination does not contain client certificate authentication, it is saved as a single configuration file.
 - If the destination provides client certificate data, it is saved as an archive, which contains the main configuration file and a JKS file.

Related Information

[Edit and Delete Destinations \[page 75\]](#)

[Destination Examples \[page 68\]](#)

1.1.3.1.2 Destination Service REST API

Destination service REST API specification for the SAP Cloud Foundry environment.

The Destination service provides a REST API that you can use to read and manage resources like destinations, certificates and destination fragments on all available levels. This API is documented in the [SAP Business Accelerator Hub](#).

It shows all available endpoints, the supported operations, parameters, possible error cases and related status codes, etc. You can also execute requests using the credentials (for example, the service key) of your Destination service instance, see [Create and Bind a Destination Service Instance \[page 209\]](#).

Related Information

[Calling the Destination Service REST API \[page 256\]](#)

["Find Destination" Response Structure \[page 259\]](#)

1.1.3.1.3 Create Destinations Using the MTA Descriptor

Use the multitarget-application (MTA) descriptor to manage destinations for complex deployments.

Content

- [Concept \[page 81\]](#)
- [Content Deployment \[page 81\]](#)
- [Create a Destination on Service Instance Creation \[page 86\]](#)

Concept

When modeling a multitarget application (MTA), you can create and update destinations from your MTA descriptor.

For more information on MTA, see [Multitarget Applications in the Cloud Foundry Environment](#).

Back to [Content \[page 81\]](#)

Content Deployment

When modeling MTAs, you can configure content deployments (for more information, see [Deploying Content with Generic Application Content Deployment](#)). The Destinations service supports such content deployments, which lets you create or update destinations by modeling them in the MTA descriptor. Other operations, like deleting a destination, are not supported by this method.

Parameters

The parameters of the content deployment have the following structure:

```
content:
  subaccount:
    existing_destinations_policy: <policy> # optional, default value is "fail".
    See "Existing destinations policy" for more details
```



```

destinations:
  - <destination descriptor 1> # See "Modeling options" to learn about the
    structure of this descriptor
    ...
  - <destination descriptor N> # See "Modeling options" to learn about the
    structure of this descriptor
instance:
  existing_destinations_policy: <policy> # optional, default value is "fail".
  See "Existing destinations policy" for more details
  destinations:
    - <destination descriptor 1> # See "Modeling options" to learn about the
      structure of this descriptor
      ...
    - <destination descriptor N> # See "Modeling options" to learn about the
      structure of this descriptor

```

📌 Note

Both the `subaccount` and `instance` sections are optional. They can both be present at the same time, or only one of them. They define the level on which the resulting destination is created.

Existing Destinations Policy

The `existing_destinations_policy` setting allows you to control what happens if a destination with the same name already exists. The possible values are:

- `fail`: Treat it as an error situation and fail the deployment. This is the default value of the setting.
- `ignore`: Keep what is currently saved in the Destination service, and skip deployment for this destination.
- `update`: Override what is currently saved in the Destination service.

Modeling Options

The `destinations` section represents an array of destination descriptors. Each of these array elements is converted to a destination and saved in the service on the respective level, based on the existing destination policy. The following options are available for modeling a destination descriptor via content deployment. They can be combined:

Destination Pointing to a Service Instance

This option lets you:

- Reference a service instance and a service key
- Specify a destination name
- Enter a description for the resulting destination (optional)
- Add additional properties and override default properties (optional)

As a result, a destination is created, based on the properties in the referenced service key.

📌 Note

This function is equivalent to the [Destinations Pointing to Service Instances \[page 71\]](#) template.

⚠ Caution

This feature is applicable for a selected set of the most commonly used services (from a Destination service perspective). If you would like to use this feature for a service which is not yet supported, let us know by opening a support ticket, see [Connectivity Support \[page 869\]](#).

In the meantime, you can follow the steps described in [Create Destinations from Scratch \[page 63\]](#).

The descriptor may have the following structures:

Without client-provided service credentials:

```
Name: <name> # name of the destination
Description: <description> # optional, a description for the destination
Authentication: <auth> # optional for some services (the default is service
specific). Some services require the Authentication to be specified, like the
workflow service. The allowed authentication types are also service-specific
ServiceInstanceName: <instance name> # the name of the service instance to which
the destination will be created
ServiceKeyName: <service key name> # the service key of the instance targeted by
ServiceInstanceName which will be used as the source for the destination values
AdditionalProp1: <value1> # optional
...
AdditionalPropN: <valueN> # optional
```

Applicable for:

- Service bindings of type "x509" which contain both a public and private key in the service key credentials
- Service bindings of type "clientsecret"

With client-provided service credentials:

```
Name: <name> # name of the destination
Description: <description> # optional, a description for the destination
Authentication: <auth> # optional for some services (the default is service
specific). Some services require the Authentication to be specified, like the
workflow service. The allowed authentication types are also service-specific
ServiceInstanceName: <instance name> # the name of the service instance to which
the destination will be created
ServiceKeyName: <service key name> # the service key of the instance targeted by
ServiceInstanceName which will be used as the source for the destination values
ServiceCredentials: # the credentials of the service key which will be provided
by the client
  PrivateKey: <value>
  ...
AdditionalProp1: <value1> # optional
...
AdditionalPropN: <valueN> # optional
```

Applicable for:

- Service bindings of type "x509" which contain only the public key in the service key credentials

Destination Pointing to a Resource Protected by an XSUAA Service Instance

This option lets you:

- Reference a service instance and a service key

- Specify a destination name
- Enter a description for the resulting destination (optional)
- Specify the URL of the target resource
- Add additional properties and override default properties (optional)

As a result, a destination is created with the token service configuration based on the properties in the referenced service key, while the URL will be the one specified when modeling the destination.

The descriptor may have the following structures:

Without client-provided service credentials:

```
Name: <name> # name of the destination
Description: <description> # optional, a description for the destination
Authentication: <auth> # optional for some services (the default is service
specific). Some services require the Authentication to be specified, like the
workflow service. The allowed authentication types are also service-specific
URL: <url> # the URL of the target resource
TokenServiceInstanceName: <instance name> # the name of the service instance
used for protecting the target resource
TokenServiceKeyName: <service key name> # the service key of the instance
targeted by ServiceInstanceName which will be used as the source for the token
service values in the destination
AdditionalProp1: <value1> # optional
...
AdditionalPropN: <valueN> # optional
```

Applicable for:

- Service bindings of type "x509" which contain both a public and private key in the service key credentials
- Service bindings of type "clientsecret"

With client-provided service credentials:

```
Name: <name> # name of the destination
Description: <description> # optional, a description for the destination
Authentication: <auth> # optional for some services (the default is service
specific). Some services require the Authentication to be specified, like the
workflow service. The allowed authentication types are also service-specific
URL: <url> # the URL of the target resource
TokenServiceInstanceName: <instance name> # the name of the service instance
used for protecting the target resource
TokenServiceKeyName: <service key name> # the service key of the instance
targeted by ServiceInstanceName which will be used as the source for the token
service values in the destination
TokenServiceCredentials: # the credentials of the service key which will be
provided by the client
  PrivateKey: <value>
AdditionalProp1: value1 # optional
...
AdditionalPropN: valueN # optional
```

Applicable for:

- Service bindings of type "x509" which contain only the public key in the service key credentials

Example:

```
_schema-version: "3.2"
ID: example
version: 0.0.1
modules:
- name: myapp
  path: ./myapp
  type: javascript.nodejs
  requires:
  - name: xsuaa_service
  provides:
  - name: myapp-route
    properties:
      url: ${default-url} #generated during deployment
- name: destination-content
  type: com.sap.application.content
  requires:
  - name: xsuaa_service
    parameters:
      service-key:
        name: xsuaa_service-key
  - name: destination-service
    parameters:
      content-target: true
  - name: myapp-route
build-parameters:
  no-source: true
parameters:
  content:
    subaccount:
      existing_destinations_policy: update
      destinations:
      - Name: myappOAuth
        URL: ~{myapp-route/url}
        Authentication: OAuth2ClientCredentials
        TokenServiceInstanceName: xsuaa_service
        TokenServiceKeyName: xsuaa_service-key
        myAdditionalProp: myValue
      - Name: workflowOAuthJwtBearer
        Authentication: OAuth2JWTBearer
        ServiceInstanceName: workflow_service
        ServiceKeyName: workflow_service-key
    instance:
      existing_destinations_policy: update
      destinations:
      - Name: workflowBasicAuthentication
        Authentication: BasicAuthentication
        ServiceInstanceName: workflow_service
        ServiceKeyName: workflow_service-key
        myAdditionalProp: myValue
resources:
- name: xsuaa_service
  type: org.cloudfoundry.managed-service
  parameters:
    service: xsuaa
    service-name: xsuaa_service
    service-plan: application
  config:
    xsappname: "myApp"
- name: workflow_service
  type: org.cloudfoundry.managed-service
  parameters:
    service: workflow
    service-name: workflow_service
    service-plan: lite
- name: destination-service
```

```
type: org.cloudfoundry.managed-service
parameters:
  service: destination
  service-name: my-destination-service
  service-plan: lite
```

Back to [Content \[page 81\]](#)

Create a Destination on Service Instance Creation

The MTA descriptor lets you create service instances and provide a JSON configuration for this operation. You can use this functionality to create a Destination service instance with a JSON, and include the required data to create or update destinations.

For more details, see [Use a Config.JSON to Create or Update a Destination Service Instance \[page 211\]](#).

Back to [Content \[page 81\]](#)

1.1.3.1.4 Create Destinations on Service Instance Creation

Use a JSON to create or update a destination when creating a Destination service instance.

When creating or updating a service instance of the Destination service, you can provide a JSON object with various configurations. One of the sections of this JSON lets you create or update destinations. Other operations, like deleting a destination, are not supported by this method.

For more information, see [Use a Config.JSON to Create or Update a Destination Service Instance \[page 211\]](#).

1.1.3.2 HTTP Destinations

Find information about HTTP destinations for Internet and on-premise connections (Cloud Foundry environment).

Destination Levels

The runtime tries to resolve a destination in the order: *Subaccount Level* → *Service Instance Level*.

Destinations for Subscribed Applications

In subscription-based scenarios, it is not always possible to consume a service instance by binding it to your application. In this case, you must create a destination pointing to your service instance. For more information, see [Destinations Pointing to Service Instances \[page 71\]](#).

Proxy Types

The proxy types supported by the Connectivity service are:

- **Internet** - The application can connect to an external REST or SOAP service on the Internet.
- **OnPremise** - The application can connect to an on-premise backend system through the Cloud Connector.
- **PrivateLink** - Selected SAP BTP services can establish a private connection with selected services in your own IaaS provider accounts.
For more information, see [What Is SAP Private Link Service?](#).

The proxy type used for a destination is specified by the destination property `ProxyType`. The default value is **Internet**.

Proxy Settings for Your Local Runtime

If you work in your local development environment behind a proxy server and want to use a service from the Internet, you need to configure your proxy settings on JVM level. To do this, proceed as follows:

1. On the [Servers](#) view, double-click the added server and choose [Overview](#) to open the editor.
2. Click the [Open Launch Configuration](#) link.
3. Choose the [\(x\)=Arguments](#) tab page.
4. In the [VM Arguments](#) box, add the following row:

```
-Dhttp.proxyHost=yourproxyHost -Dhttp.proxyPort=yourProxyPort  
-Dhttps.proxyHost=yourproxyHost -Dhttps.proxyPort=yourProxyPort
```

5. Choose [OK](#).
6. Start or restart your SAP HANA Cloud local runtime.

Configuring Authentication

When creating an HTTP destination, you can use different authentication types for access control:

- [Server Certificate Authentication \[page 89\]](#)
- [Principal Propagation SSO Authentication for HTTP \[page 92\]](#)
- [OAuth SAML Bearer Assertion Authentication \[page 94\]](#)

- [Client Authentication Types for HTTP Destinations \[page 101\]](#)
- [OAuth Client Credentials Authentication \[page 104\]](#)
- [OAuth User Token Exchange Authentication \[page 112\]](#)
- [OAuth Password Authentication \[page 118\]](#)
- [OAuth JWT Bearer Authentication \[page 122\]](#)
- [SAML Assertion Authentication \[page 128\]](#)
- [OAuth with X.509 Client Certificates \[page 133\]](#)
- [OAuth Refresh Token Authentication \[page 134\]](#)
- [OAuth Authorization Code Authentication \[page 139\]](#)
- [OAuth Technical User Propagation Authentication \[page 145\]](#)
- [Using Client Assertion with OAuth Flows \[page 152\]](#)

Custom Query Parameters and Headers

By default, the Destination service does not use URL-associated queries and header parameters.

For most authentication types however, you can add them as custom parameters to the URL of a destination.

→ Tip

You can use tools like the *transparent proxy for Kubernetes* to process those attributes automatically at runtime.

For more information, see [Using the Transparent Proxy \[page 802\]](#).

When using the *Destinations* editor for destination configuration, you can add these parameters as *additional properties*:

- `URL.headers.HEADER_KEY` for headers
- `URL.queries.QUERY_KEY` for query parameters

Replace `HEADER_KEY` and `QUERY_KEY` with the name of the headers or query parameters and set the respective values.

Destination Configuration Blank Template Service Instance

Name: *	MY-DESTINATION
Type:	HTTP
Description:	
URL: *	https://example.com
Proxy Type:	Internet
Authentication:	NoAuthentication

Additional Properties		New Property
URL.queries.language	EN	
URL.headers.apiKey	<my-api-key>	

Use default JDK truststore

In the example above, the destination has a `language` query parameter with the value `EN` and an `apiKey` header with the value `<my-api-key>`.

Those additional headers and query parameters are added for every communication with the given destination.

For more information on additional properties, see [Create HTTP Destinations \[page 63\]](#) (step 9) and the details of the respective [authentication type \[page 87\]](#).

Related Information

[OAuth with X.509 Client Certificates \[page 133\]](#)

[Create HTTP Destinations \[page 63\]](#)

1.1.3.2.1 Server Certificate Authentication

Create and configure a *Server Certificate* destination for an application in the Cloud Foundry environment.

Context

The server certificate validation is applicable to all destinations with proxy type `Internet` and `PrivateLink` that use the HTTPS protocol.

Note

TLS 1.2 became the default TLS version of HTTP destinations. If an HTTP destination is consumed by a java application the change will be effective after restart. All HTTP destinations that use the HTTPS protocol and have `ProxyType=Internet` can be affected. Previous TLS versions can be used by configuring an additional property `TLSVersion=TLSv1.0` or `TLSVersion=TLSv1.1`.

Properties

Property	Description
<code>TLSVersion</code>	Optional property. Can be used to specify the preferred TLS version to be used by the current destination. Since TLS 1.2 is not enabled by default on the older java versions this property can be used to configure TLS 1.2 in case this is required by the server configured in this destination. It is usable only in HTTP destinations. Example: <code>TLSVersion=TLSv1.2</code> .

Property	Description
TrustStoreLocation	<p>Path to the keystore file which contains trusted certificates (Certificate Authorities) for authentication against a remote client.</p> <p>To find the allowed keystore file formats, see Use Destination Certificates [page 76].</p> <ol style="list-style-type: none"> 1. The relative path to the keystore file. The root path is the server's location on the file system. 2. The name of the keystore file.
	<p>Note</p> <p>If the TrustStoreLocation property is not specified, the JDK trust store is used as a default trust store for the destination.</p>
TrustStorePassword	<p>Password for the JKS trust store file. This property is mandatory in case TrustStoreLocation is used.</p>
TrustAll	<p>If this property is set to TRUE in the destination, the server certificate will not be checked for SSL connections. It is intended for test scenarios only, and should not be used in production (since the SSL server certificate is not checked, the server is not authenticated). The possible values are TRUE or FALSE; the default value is FALSE (that is, if the property is not present at all).</p> <p>In case TrustAll = TRUE, the TrustStoreLocation property is ignored so you can omit it.</p> <p>In case <TrustAll> = FALSE, the <TrustStoreLocation> property is mandatory to be used.</p>

Property	Description
HostnameVerifier	<p>Optional property. It has two values: Strict and BrowserCompatible. This property specifies how the server hostname matches the names stored inside the server's X.509 certificate. This verifying process is only applied if TLS or SSL protocols are used and is not applied if the TrustAll property is specified. The default value (used if no value is explicitly specified) is Strict.</p> <ul style="list-style-type: none"> • Strict <code>HostnameVerifier</code> works in the same way as Oracle Java 1.4, Oracle Java 5, and Oracle Java 6-rc. It is also similar to Microsoft Internet Explorer 6. This implementation appears to be compliant with RFC 2818 for dealing with wildcards. A wildcard such as <code>*.foo.com</code> matches only subdomains at the same level, for example <code>a.foo.com</code>. It does not match deeper subdomains such as <code>a.b.foo.com</code>. • BrowserCompatible <code>HostnameVerifier</code> works in the same way as Curl and Mozilla Firefox. The hostname must match either the first common name (CN) or any of the subject-alts. A wildcard can occur in the CN and in any of the subject-alts. <p>The only difference between BrowserCompatible and Strict is that a wildcard (such as <code>.foo.com</code>) with BrowserCompatible matches all subdomains, including <code>a.b.foo.com</code>.</p> <p>For more information about these Java classes, see Package org.apache.http.conn.ssl.</p> <p>In case <code><TrustAll> = TRUE</code>, the <code><HostnameVerifier></code> property is ignored so you can omit it.</p>

Note

You can upload trust store JKS files using the same command as for uploading destination configuration property files. You only need to specify the JKS file instead of the destination configuration file.

Note

Connections to remote services which require *Java Cryptography Extension (JCE) unlimited strength jurisdiction policy* are not supported.

Configuration

- [Managing Destinations \[page 59\]](#)

Related Information

[Client Authentication Types for HTTP Destinations \[page 101\]](#)

1.1.3.2.2 Principal Propagation SSO Authentication for HTTP

Forward the identity of a cloud user from a Cloud Foundry application to a backend system via HTTP to enable single sign-on (SSO).

Context

A *PrincipalPropagation* destination enables single sign-on (SSO) by forwarding the identity of a cloud user to the Cloud Connector, and from there to the target on-premise system. In this way, the cloud user's identity can be provided without manual logon.

Note

This authentication type applies only for on-premise connectivity.

Configuration Steps

You can create and configure a *PrincipalPropagation* destination by using the properties listed below, and deploy it on SAP BTP. For more information, see [Managing Destinations \[page 59\]](#).

Properties

The following credentials need to be specified:

Property	Description
Name	Destination name. Must be unique for the destination level.
Type	Destination type. Use HTTP for all HTTP(S) destinations.
URL	Virtual URL of the protected on-premise application.
Authentication	Authentication type. Use PrincipalPropagation as value.
ProxyType	You can only use proxy type OnPremise .

Property	Description
CloudConnectorLocationId	You can connect multiple Cloud Connectors to a subaccount as long as their location ID is different. The location ID specifies the Cloud Connector over which the connection is opened. The default value is an empty string identifying the Cloud Connector that is connected without any location ID.

URL.headers.<header-key>	A static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a URL.headers prefix separated by dot-delimiter. For example:
--------------------------	---

Sample Code

```
{
  ...
  "URL.headers.<header-key-1>" :
  "<header-value-1>",
  ...
  "URL.headers.<header-key-N>" :
  "<header-value-N>",
}
```

Note

This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.

URL.queries.<query-key>	A static key prefix used as a namespace grouping of URL's query parameters whose values will be sent to the target endpoint. For each query parameter's key, you must add a URL.queries prefix separated by dot-delimiter. For example:
-------------------------	---

Sample Code

```
{
  ...
  "URL.queries.<query-key-1>" :
  "<query-value-1>",
  ...
  "URL.queries.<query-key-N>" :
  "<query-value-N>",
}
```

Property	Description
	<p data-bbox="826 367 927 394">📌 Note</p> <p data-bbox="826 423 1342 656">This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p>

Example

```
Name=OnPremiseDestination
Type=HTTP
URL= http://virtualhost:80
Authentication=PrincipalPropagation
ProxyType=OnPremise
```

Related Information

[Principal Propagation \[page 168\]](#)

1.1.3.2.3 OAuth SAML Bearer Assertion Authentication

Create and configure an *OAuth SAML Bearer Assertion* destination for an application in the Cloud Foundry environment.

Context

You can call an OAuth2-protected remote system/API and propagate a user ID to the remote system by using the `OAuth2SAMLBearerAssertion` authentication type. The Destination service provides functionality for automatic token retrieval and caching, by automating the construction and sending of the SAML assertion. This simplifies application development, leaving you with only constructing the request to the remote system by providing the token, which is fetched for you by the Destination service. For more information, see [User Propagation via SAML 2.0 Bearer Assertion Flow \[page 251\]](#).

Properties

The table below lists the destination properties for *OAuth2SAMLBearerAssertion* authentication type. You can find the values for these properties in the provider-specific documentation of OAuth-protected services. Usually, only a subset of the optional properties is required by a particular service provider.

Property	Description
Required	
Name	Name of the destination. Must be unique for the destination level.
Type	Destination type. Choose HTTP for all HTTP(S) destinations.
URL	URL of the target endpoint.
ProxyType	You can only use proxy type Internet or OnPremise . If OnPremise is used, the OAuth server must be accessed through the Cloud Connector.
Authentication	Authentication type. Use OAuth2SAMLBearerAssertion as value.
KeyStoreLocation	Contains the name of the certificate configuration to be used for <i>per-destination</i> SAML assertion signing. This certificate will be used instead of the standard subaccount-wide signing key. For more information, see Set up Trust Between Systems [page 172] .
KeyStorePassword	Contains the password for the certificate configuration (if one is needed) when using a <i>per-destination</i> SAML assertion signing certificate.
audience	Intended audience for the assertion, which is verified by the OAuth authorization server. For more information, see SAML 2.0 Bearer Assertion Profiles for OAuth 2.0 .
clientKey	Key that identifies the consumer to the authorization server.

Property	Description
tokenServiceURL	<p>The URL of the token service, against which the token exchange is performed. Depending on the <code>Token Service URL</code> type, this property is interpreted in different ways during the automatic token retrieval:</p> <ul style="list-style-type: none"> For Dedicated, the token service URL is taken as is. For Common, the token service URL is searched for the tenant placeholder <code>{tenant}</code>. <code>{tenant}</code> is resolved as the subdomain of the subaccount on behalf of which the caller is performing the call. If the placeholder is not found, <code>{tenant}</code> is inserted as a subdomain of the token service URL. The subaccount subdomain is mandated during creation of the subaccount, see Create a Subaccount. <p>Examples of interpreting the token service URL for the token service URL type Common, if the call to the Destination service is on behalf of a subaccount subdomain with value <code>mytenant</code>:</p> <ul style="list-style-type: none"> <code>https://authentication.us10.hana.ondemand.com/oauth/token</code> → <code>https://mytenant.authentication.us10.hana.ondemand.com/oauth/token</code> <code>https://{tenant}.authentication.us10.hana.ondemand.com/oauth/token</code> → <code>https://mytenant.authentication.us10.hana.ondemand.com/oauth/token</code> <code>https://authentication.myauthserver.com/tenant/{tenant}/oauth/token</code> → <code>https://authentication.myauthserver.com/tenant/mytenant/oauth/token</code> <code>https://oauth.{tenant}.myauthserver.com/token</code> → <code>https://oauth.mytenant.myauthserver.com/token</code>
tokenServiceURLType	Either Dedicated - if the token service URL serves only a single tenant, or Common - if the token service URL serves multiple tenants.
tokenServiceUser	User for basic authentication to OAuth server (if required).
tokenServicePassword	Password for <code>tokenServiceUser</code> (if required).

Property	Description
(Deprecated) <code>SystemUser</code>	<p>User to be used when requesting an access token from the OAuth authorization server. If this property is not specified, the currently logged-in user is used.</p> <div style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>⚠ Caution</p> <p>This property is deprecated and will be removed soon. We recommend that you work on behalf of specific (named) users instead of working with a technical user.</p> <p>As an alternative for technical user communication, we strongly recommend that you use one of these authentication types:</p> <ul style="list-style-type: none"> • Basic Authentication (see Client Authentication Types for HTTP Destinations [page 101]) • Client Certificate Authentication (see Client Authentication Types for HTTP Destinations [page 101]) • OAuth Client Credentials Authentication [page 104] <p>To extend an OAuth access token's validity, consider using an OAuth refresh token.</p> </div>
<code>authnContextClassRef</code>	Value of the <code>AuthnContextClassRef</code> tag, which is part of generated OAuth2SAMLBearerAssertion authentication. For more information, see SAML 2.0 specification .
Additional	
<code>nameQualifier</code>	Security domain of the user for which access token is requested.
<code>companyId</code>	Company identifier.
<code>assertionIssuer</code>	Issuer of the SAML assertion.
<code>assertionRecipient</code>	Recipient of the SAML assertion. If not set, the token service URL will be the assertion's recipient.
<code>nameIdFormat</code>	Value of the <code>NameIdFormat</code> tag, which is part of generated OAuth2SAMLBearerAssertion authentication. For more information, see SAML 2.0 specification .
<code>userIdSource</code>	When this property is set, the generated SAML2 assertion uses the currently logged-in user as a value for the <code>NameId</code> tag. See User Propagation via SAML 2.0 Bearer Assertion Flow [page 251] .
<code>scope</code>	The value of the OAuth 2.0 scope parameter, expressed as a list of space-delimited, case-sensitive strings.

Property	Description
<code>tokenServiceURL.headers.<header-key></code>	<p>A static key prefix used as a namespace grouping of the <code>tokenServiceURL</code>'s HTTP headers. Its values will be sent to the token service during token retrieval. For each HTTP header's key you must add a <code>'tokenServiceURL.headers'</code> prefix separated by dot delimiter. For example:</p> <div data-bbox="826 555 1396 913" data-label="Code-Block"> <p>↔ Sample Code</p> <pre>{ ... "tokenServiceURL.headers.<header-key-1>" : "<header-value-1>", ... "tokenServiceURL.headers.<header-key-N>" : "<header-value-N>", }</pre> </div>
<code>tokenServiceURL.ConnectionTimeoutInSeconds</code>	<p>Defines the connection timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 60 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.</p>
<code>tokenServiceURL.SocketReadTimeoutInSeconds</code>	<p>Defines the read timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 600 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.</p>
<code>tokenServiceURL.queries.<query-key></code>	<p>A static key prefix used as a namespace grouping of <code>tokenServiceURL</code>'s query parameters. Its values will be sent to the token service during token retrieval. For each query parameter's key you must add a <code>'tokenServiceURL.queries'</code> prefix separated by dot delimiter. For example:</p> <div data-bbox="826 1451 1396 1809" data-label="Code-Block"> <p>↔ Sample Code</p> <pre>{ ... "tokenServiceURL.queries.<query-key-1>" : "<query-value-1>", ... "tokenServiceURL.queries.<query-key-N>" : "<query-value-N>", }</pre> </div>

Property	Description
<code>tokenService.body.<param-key></code>	<p>A static key prefix used as a namespace grouping of parameters which are sent as part of the token request to the token service during token retrieval. For each request, a <code>tokenService.body</code> prefix must be added to the parameter key, separated by dot-delimiter. For example:</p> <div data-bbox="826 539 1043 573" data-label="Section-Header"> <p>↔ Sample Code</p> </div> <pre data-bbox="842 607 1342 815"> { ... "tokenService.body.<param-key-1>" : "<param-value-1>", ... "tokenService.body.<param-key-N>" : "<param-value-N>", }</pre>

Property	Description
URL.queries.<query-key>	<p>A static key prefix used as a namespace grouping of URL's query parameters whose values will be sent to the target endpoint. For each query parameter's key, you must add a URL.queries prefix separated by dot-delimiter. For example:</p> <div data-bbox="826 539 1043 573" style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"> <p>Sample Code</p> <pre data-bbox="842 607 1326 815"> { ... "URL.queries.<query-key-1>" : "<query-value-1>", ... "URL.queries.<query-key-N>" : "<query-value-N>", }</pre> </div>

Note

This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.

Restriction

If the value is a private endpoint (for example, *localhost*), the Destination service will not be able to perform the verification of the *X-User-Token* header when using the "Find Destination" API.

Example

The connectivity destination below provides HTTP access to the OData API of the SuccessFactors Jam.

```
URL=https://demo.sapjam.com/OData/OData.svc
Name=sap_jam_odata
ProxyType=Internet
Type=HTTP
Authentication=OAuth2SAMLBearerAssertion
tokenServiceURL=https://demo.sapjam.com/api/v1/auth/token
clientKey=<unique_generated_string>
audience=cubetree.com
nameQualifier=www.successfactors.com
apiKey=<apiKey>
```

The response for "find destination" contains an `authTokens` object in the format given below. For more information on the fields in `authTokens`, see "[Find Destination](#)" [Response Structure](#) [page 259].

Sample Code

```
"authTokens": [
  {
    "type": "Bearer",
    "value": "eyJhbGciOiJSUzI1NiIsInR5cCI6IiJ0eS...",
    "http_header": {
      "key": "Authorization",
      "value": "Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IiJ0eS..."
    }
  }
]
```

Related Information

[Create HTTP Destinations](#) [page 63]

[Destination Examples](#) [page 68]

[User Propagation via SAML 2.0 Bearer Assertion Flow](#) [page 251]

[User Propagation between Cloud Foundry Applications](#) [page 188]

[Exchanging User JWTs via OAuth2UserTokenExchange Destinations](#) [page 263]

1.1.3.2.4 Client Authentication Types for HTTP Destinations

Find details about client authentication types for HTTP destinations in the Cloud Foundry environment.

Context

This section lists the supported client authentication types and the relevant supported properties.

No Authentication

This authentication type is used for destinations that refer to a service on the Internet, an on-premise system, or a *Private Link* endpoint that does not require authentication. The relevant property value is:

```
Authentication=NoAuthentication
```

Note

When a destination is using HTTPS protocol to connect to a Web resource, the JDK truststore is used as truststore for the destination.

Basic Authentication

Used for destinations that refer to a service on the Internet, an on-premise system, or a *Private Link* endpoint that requires basic authentication. The relevant property value is:

```
Authentication=BasicAuthentication
```

The following credentials need to be specified:

Caution

Do not use your own *personal credentials* in the `<User>` and `<Password>` fields. Always use a *technical user* instead.

Property	Description
User	User name of the technical user to be used.
Password	Password of the technical user to be used.
Preemptive	If this property is not set or is set to TRUE (that is, the default behavior is to use preemptive sending), the authentication token is sent preemptively. Otherwise, it relies on the challenge from the server (401 HTTP code). The default value (used if no value is explicitly specified) is TRUE . For more information about preemptiveness, see http://tools.ietf.org/html/rfc2617#section-3.3 .

Note

When a destination is using the HTTPS protocol to connect to a Web resource, the JDK truststore is used as truststore for the destination.

Note

Basic Authentication and **No Authentication** can be used in combination with `ProxyType=OnPremise`. In this case, also the `CloudConnectorLocationId` property can be specified. You can connect multiple Cloud Connectors to a subaccount as long as their location ID is different. The value defines the location ID identifying the Cloud Connector over which the connection shall be opened. The default value is the empty string identifying the Cloud Connector that is connected without any location ID.

The response for "find destination" contains an `authTokens` object in the format given below. For more information on the fields in `authTokens`, see ["Find Destination" Response Structure \[page 259\]](#).

Sample Code

```
"authTokens": [
  {
    "type": "Basic",
    "value": "dGVzdDpwYXNzMTIzNDU=",
    "http_header": {
      "key": "Authorization",
      "value": "Basic dGVzdDpwYXNzMTIzNDU="
    }
  }
]
```

Client Certificate Authentication

Used for destinations that refer to a service on the Internet or a *Private Link* endpoint. The relevant property value is:

Authentication=**ClientCertificateAuthentication**

The following credentials need to be specified:

Property	Description
<code>KeyStore.Source</code>	<p>Optional. Specifies the storage location of the certificate to be used by the client. Supported values are:</p> <ul style="list-style-type: none"> <code>ClientProvided</code>: The key store is managed by the client (the application itself). <code>DestinationService</code>: The key store is managed by the Destination service. <p>If the property is not set, the key store is managed by the Destination service (default).</p>
<code>KeyStoreLocation</code>	<p>The name of the key store file that contains the client certificate(s) for client certificate authentication against a remote server. This property is optional if <code>KeyStore.Source</code> is set to <code>ClientProvided</code>.</p>
<code>KeyStorePassword</code>	<p>Password for the key store file specified by <code>KeyStoreLocation</code>. This property is optional if <code>KeyStoreLocation</code> is used in combination with <code>KeyStore.Source</code>, and <code>KeyStore.Source</code> is set to <code>ClientProvided</code>.</p>

Note

You can upload `keyStore` JKS files using the same command for uploading destination configuration property file. You only need to specify the JKS file instead of the destination configuration file.

Configuration

- [Managing Destinations \[page 59\]](#)

Related Information

[Server Certificate Authentication \[page 89\]](#)

1.1.3.2.5 OAuth Client Credentials Authentication

Create and configure an `OAuth2ClientCredentials` destination to consume OAuth-protected resources from a Cloud Foundry application.

SAP BTP supports applications to use the OAuth client credentials flow for consuming OAuth-protected resources.

The client credentials are used to request an access token from an OAuth authorization server.

Note

The retrieved access token is cached and auto-renovated. When a token is about to expire, a new token is created shortly before the expiration of the old one.

Configuration Steps

You can create and configure an *OAuth2ClientCredentials* destination using the properties listed below, and deploy it on SAP BTP. To create and configure a destination, follow the steps described in [Managing Destinations \[page 59\]](#).

Properties

The table below lists the destination properties required for the *OAuth2ClientCredentials* authentication type.

Property	Description
Required	
Name	Destination name. Must be unique for the destination level.
Type	Destination type. Use HTTP as value for all HTTP(S) destinations.
URL	URL of the protected resource on the called application.
ProxyType	You can only use proxy type Internet or OnPremise . If OnPremise is used, the OAuth server must be accessed through the Cloud Connector.
Authentication	Authentication type. Use OAuth2ClientCredentials as value.
clientId	Client ID used to retrieve the access token.
clientSecret	Client secret for the Client ID.

Property	Description
tokenServiceURL	<p>URL of the token service, against which token retrieval is performed. Depending on the tokenServiceURLType, this property is interpreted in different ways during automatic token retrieval:</p> <ul style="list-style-type: none"> For Dedicated, the tokenServiceURL is used as is. For Common, the tokenServiceURL is searched for the tenant placeholder <i>{tenant}</i>. It is resolved as subdomain of the subaccount on whose behalf the caller is performing the call to the Destination service API for fetching the destination. If the placeholder is not found, <i>{tenant}</i> is processed as a subdomain of the tokenServiceURL. See the Destination service REST API to learn how the subaccount's subdomain is specified. The subaccount subdomain is mandated during creation of the subaccount, see Create a Subaccount (SAP BTP Core documentation). <p>Examples of interpreting of the tokenServiceURL for tokenServiceURLType Common, if the call to the Destination service is done on behalf of a subaccount with subdomain value <i>mytenant</i>:</p> <ul style="list-style-type: none"> https://authentication.us10.hana.ondemand.com/oauth/token → https://mytenant.authentication.us10.hana.ondemand.com/oauth/token https://{tenant}.authentication.us10.hana.ondemand.com/oauth/token → https://mytenant.authentication.us10.hana.ondemand.com/oauth/token https://authentication.myauthserver.com/tenant/{tenant}/oauth/token → https://authentication.myauthserver.com/tenant/mytenant/oauth/token https://oauth.{tenant}.myauthserver.com/token → https://oauth.mytenant.myauthserver.com/token
tokenServiceURLType	Either Dedicated (if the tokenServiceURL serves only a single tenant), or Common (if the tokenServiceURL serves multiple tenants).
tokenServiceUser	User for basic authentication to OAuth server (if required).
tokenServicePassword	Password for tokenServiceUser (if required).
Additional	
scope	The value of the OAuth 2.0 scope parameter expressed as a list of space-delimited, case-sensitive strings.

Property	Description
<code>tokenServiceURL.headers.<header-key></code>	<p>A static key prefix used as a namespace grouping of the <code>tokenServiceURL</code>'s HTTP headers. Its values will be sent to the token service during token retrieval. For each HTTP header's key you must add a <code>'tokenServiceURL.headers'</code> prefix separated by dot delimiter. For example:</p> <div data-bbox="826 555 1396 913" style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>↔ Sample Code</p> <pre> { ... "tokenServiceURL.headers.<header-key-1>" : "<header-value-1>", ... "tokenServiceURL.headers.<header-key-N>" : "<header-value-N>", } </pre> </div>
<code>tokenServiceURL.ConnectionTimeoutInSeconds</code>	<p>Defines the connection timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 60 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.</p>
<code>tokenServiceURL.SocketReadTimeoutInSeconds</code>	<p>Defines the read timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 600 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.</p>
<code>tokenServiceURL.queries.<query-key></code>	<p>A static key prefix used as a namespace grouping of <code>tokenServiceURL</code>'s query parameters. Its values will be sent to the token service during token retrieval. For each query parameter's key you must add a <code>'tokenServiceURL.queries'</code> prefix separated by dot delimiter. For example:</p> <div data-bbox="826 1460 1396 1818" style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>↔ Sample Code</p> <pre> { ... "tokenServiceURL.queries.<query-key-1>" : "<query-value-1>", ... "tokenServiceURL.queries.<query-key-N>" : "<query-value-N>", } </pre> </div>

Property	Description
<code>tokenService.body.<param-key></code>	<p>A static key prefix used as a namespace grouping of parameters which are sent as part of the token request to the token service during token retrieval. For each request, a <code>tokenService.body</code> prefix must be added to the parameter key, separated by dot-delimiter. For example:</p> <div data-bbox="804 528 1394 853" style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 10px; margin-top: 10px;"> <p>Sample Code</p> <pre> { ... "tokenService.body.<param-key-1>" : "<param-value-1>", ... "tokenService.body.<param-key-N>" : "<param-value-N>", } </pre> </div>
<code>tokenService.KeyStoreLocation</code>	<p>Contains the name of the certificate configuration to be used. This property is required when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133].</p>
<code>tokenService.KeyStorePassword</code>	<p>Contains the password for the certificate configuration (if one is needed) when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133].</p>
<code>tokenService.addClientCredentialsInBody</code>	<p>Specifies whether the client credentials should be placed in the request body of the token request, rather than the Authorization header. Default is true.</p> <div data-bbox="804 1245 1394 1469" style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>If set to false, but <code>tokenServiceUser / tokenServicePassword</code> are set, <code>tokenServiceUser / tokenServicePassword</code> are taken with priority.</p> </div>
<code>clientAssertion.type</code>	<p>When using this destination as a client assertion provider, you can specify the format of the client assertion as defined by the authorization server. The supported values are:</p> <ul style="list-style-type: none"> • "urn:ietf:params:oauth:client-assertion-type:saml2-bearer" => indicating a SAML Bearer assertion. • "urn:ietf:params:oauth:client-assertion-type:jwt-bearer" => indicating a JWT Bearer token. <p>This is used in case of automated client assertion fetching by the service.</p> <p>For more information, see Client Assertion with Automated Assertion Fetching by the Service [page 154].</p>

Property	Description
clientAssertion.destinationName	<p>Name of the destination that provides client assertions when using client assertion authentication mechanism. Must be on the same subaccount or service instance as this destination. This is used in case of automated client assertion fetching by the service.</p> <p>For more information, see Client Assertion with Automated Assertion Fetching by the Service [page 154].</p>
URL.headers.<header-key>	<p>A static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a URL.headers prefix separated by dot-delimiter. For example:</p> <div data-bbox="804 786 1398 1111" data-label="Code-Block"> <p>Sample Code</p> <pre> { ... "URL.headers.<header-key-1>" : "<header-value-1>", ... "URL.headers.<header-key-N>" : "<header-value-N>", } </pre> </div> <div data-bbox="804 1137 1398 1435" data-label="Text"> <p>Note</p> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p> </div>

Property	Description
<code>URL.queries.<query-key></code>	<p>A static key prefix used as a namespace grouping of URL's query parameters whose values will be sent to the target endpoint. For each query parameter's key, you must add a <code>URL.queries</code> prefix separated by dot-delimiter. For example:</p> <div data-bbox="826 533 1396 853" data-label="Code-Block"> <pre> ❏ Sample Code { ... "URL.queries.<query-key-1>" : "<query-value-1>", ... "URL.queries.<query-key-N>": "<query-value-N>", } </pre> </div> <div data-bbox="826 887 1396 1173" data-label="Text"> <p>Note</p> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p> </div>

Note

When the OAuth authorization server is called, it accepts the trust settings of the destination, see [Server Certificate Authentication \[page 89\]](#).

When using an SAP BTP Neo OAuth service (`https://api.{landscape-domain}/oauth2/apitoken/v1?grant_type=client_credentials` or `oauthservices.{landscape-domain}/oauth2/apitoken/v1?grant_type=client_credentials`) as `TokenServiceURL`, or any other OAuth token service which accepts client credentials only as authorization header, you must set the `clientId` and `clientSecret` values also for the `tokenServiceUser` and `tokenServicePassword` properties.

Example: Neo OAuth Token Service

Sample Code

```

URL=https://api.{landscape-domain}/desired-service-path
Name=sapOAuthCC
ProxyType=Internet
Type=HTTP

```


1.1.3.2.6 OAuth User Token Exchange Authentication

Learn about the `OAuth2UserTokenExchange` authentication type for HTTP destinations in the Cloud Foundry environment: use cases, supported properties and ways to retrieve an access token in an automated way.

Content

[Overview \[page 112\]](#)

[Properties \[page 112\]](#)

[Example: AuthTokens Object Response \[page 117\]](#)

Overview

When a user is logged into an application that needs to call another application and pass the user context, the caller application must perform a user token exchange.

The user token exchange is a sequence of steps during which the initial user token is handed over to the authorization server and, in exchange, another access token is returned.

The calling application first receives a refresh token out of which the actual user access token is created. The resulting user access token contains the user and tenant context as well as technical access metadata, like scopes, that are required for accessing the target application.

Using the `OAuth2UserTokenExchange` authentication type, the Destination service performs all these steps automatically, which lets you simplify your application development in the Cloud Foundry environment.

Back to [Content \[page 112\]](#)

Properties

To configure a destination of this type, you must specify all the required properties. You can create destinations of this type via the cloud cockpit ([Access the Destinations Editor \[page 61\]](#)) or the [Destination Service REST API \[page 80\]](#).

The following table shows the required properties along with their semantics.

Field/Parameter	JSON Key	Input/Description
Required		
URL	URL	URL of the target endpoint.
Token Service URL	tokenServiceURL	<p>The URL of the token service, against which the token exchange is performed. Depending on the <code>Token Service URL Type</code>, this property is interpreted in different ways during the automatic token retrieval:</p> <ul style="list-style-type: none"> For Dedicated, the token service URL is taken as is. For Common, the token service URL is searched for the tenant placeholder <code>{tenant}</code>. <code>{tenant}</code> is resolved as the subdomain of the subaccount on behalf of which the caller is performing the call. If the placeholder is not found, <code>{tenant}</code> is inserted as a subdomain of the token service URL. See Automated Access Token Retrieval [page 264] for information about how the tenant is determined. The subaccount subdomain is mandated during creation of the subaccount, see Create a Subaccount. <p>Examples of interpreting the token service URL for the token service URL type Common, if the call to the Destination service is on behalf of a subaccount subdomain with value <code>mytenant</code>:</p> <ul style="list-style-type: none"> <code>https://authentication.us10.hana.ondemand.com/oauth/token</code> → <code>https://mytenant.authentication.us10.hana.ondemand.com/oauth/token</code> <code>https://{tenant}.authentication.us10.hana.ondemand.com/oauth/token</code> → <code>https://mytenant.authentication.us10.hana.ondemand.com/oauth/token</code> <code>https://authentication.myauthserver.com/tenant/{tenant}/oauth/token</code> → <code>https://authentication.myauthserver.com/tenant/mytenant/oauth/token</code> <code>https://oauth.{tenant}.myauthserver.com/token</code> → <code>https://oauth.mytenant.myauthserver.com/token</code>
Name	Name	Name of the destination. Must be unique for the destination level.
Description	Description	A human-readable description of the destination.
Client Secret	clientSecret	OAuth 2.0 client secret to be used for the user access token exchange.
Client ID	clientId	OAuth 2.0 client ID to be used for the user access token exchange.
Authentication	Authentication	OAuth2UserTokenExchange in this case.

Field/Parameter	JSON Key	Input/Description
Proxy Type	ProxyType	You can only use proxy type Internet or OnPremise . If OnPremise is used, the OAuth server must be accessed through the Cloud Connector.
Type	Type	Choose HTTP (for HTTP or HTTPS communication).
Token Service URL Type	tokenServiceURLType	<ul style="list-style-type: none"> Choose Dedicated if the token service URL serves only a single tenant. Choose Common if the token service URL serves multiple tenants.
Optional		
Description	Description	Description of the destination.
Additional		
	scope	The value of the OAuth 2.0 scope parameter expressed as a list of space-delimited, case-sensitive strings.
	tokenServiceURL.headers.<header-key>	<p>A static key prefix used as a namespace grouping of the tokenServiceURL's HTTP headers. Its values will be sent to the token service during token retrieval. For each HTTP header's key you must add a 'tokenServiceURL.headers' prefix separated by dot delimiter. For example:</p> <div data-bbox="576 981 1398 1301" data-label="Code-Block"> <pre> ↔ Sample Code { ... "tokenServiceURL.headers.<header-key-1>" : "<header-value-1>", ... "tokenServiceURL.headers.<header-key-N>" : "<header-value-N>", } </pre> </div>
	tokenServiceURL.queries.<query-key>	<p>A static key prefix used as a namespace grouping of tokenServiceURL's query parameters. Its values will be sent to the token service during token retrieval. For each query parameter's key you must add a 'tokenServiceURL.queries' prefix separated by dot delimiter. For example:</p> <div data-bbox="576 1491 1398 1812" data-label="Code-Block"> <pre> ↔ Sample Code { ... "tokenServiceURL.queries.<query-key-1>" : "<query-value-1>", ... "tokenServiceURL.queries.<query-key-N>" : "<query-value-N>", } </pre> </div>

Field/Parameter	JSON Key	Input/Description
<code>tokenService.body.<param-key></code>		A static key prefix used as a namespace grouping of parameters which are sent as part of the token request to the token service during token retrieval. For each request, a <code>tokenService.body</code> prefix must be added to the parameter key, separated by dot-delimiter. For example:

Sample Code

```
{
  ...
  "tokenService.body.<param-key-1>" : "<param-value-1>",
  ...
  "tokenService.body.<param-key-N>" : "<param-value-N>",
}
```

<code>URL.headers.<header-key></code>		A static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a <code>URL.headers</code> prefix separated by dot-delimiter. For example:
---	--	--

Sample Code

```
{
  ...
  "URL.headers.<header-key-1>" : "<header-value-1>",
  ...
  "URL.headers.<header-key-N>" : "<header-value-N>",
}
```

Note

This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.

Field/Parameter	JSON Key	Input/Description
	<code>URL.queries.<query-key></code>	<p>A static key prefix used as a namespace grouping of URL's query parameters whose values will be sent to the target endpoint. For each query parameter's key, you must add a <code>URL.queries</code> prefix separated by dot-delimiter. For example:</p> <div data-bbox="596 472 815 510" data-label="Section-Header"> <h4>Sample Code</h4> </div> <pre data-bbox="612 546 1355 725"> { ... "URL.queries.<query-key-1>" : "<query-value-1>", ... "URL.queries.<query-key-N>" : "<query-value-N>", } </pre> <div data-bbox="596 804 699 837" data-label="Section-Header"> <h4>Note</h4> </div> <p data-bbox="596 862 1382 994">This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p>
	<code>tokenService.KeyStoreLocation</code>	Contains the name of the certificate configuration to be used. This property is required when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133] .
	<code>tokenService.KeyStorePassword</code>	Contains the password for the certificate configuration (if one is needed) when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133] .
	<code>tokenService.addClientCredentialsInBody</code>	Specifies whether the client credentials should be placed in the request body of the token request, rather than the <code>Authorization</code> header. Default is true . <div data-bbox="596 1375 699 1408" data-label="Section-Header"> <h4>Note</h4> </div> <p data-bbox="596 1433 1366 1534">If set to false, but <code>tokenServiceUser / tokenServicePassword</code> are set, <code>tokenServiceUser / tokenServicePassword</code> are taken with priority.</p>
	<code>x_user_token.jwks</code>	<p>Base64-encoded <i>JSON web key set</i>, containing the signing keys which are used to validate the JWT provided in the <i>X-User-Token</i> header.</p> <p>For more information, see JWK Set Format.</p>

Field/Parameter	JSON Key	Input/Description
	<code>x_user_token.jwks_uri</code>	URI of the <i>JSON web key set</i> , containing the signing keys which are used to validate the JWT provided in the <i>X-User-Token</i> header.

⚠ Restriction

If the value is a private endpoint (for example, *localhost*), the Destination service will not be able to perform the verification of the X-User-Token header when using the "Find Destination" API.

For more information, see [OpenID Connect Discovery](#).

[Back to Content \[page 112\]](#)

Example: AuthTokens Object Response

The response for "find destination" contains an `authTokens` object in the format given below. For more information on the fields in `authTokens`, see ["Find Destination" Response Structure \[page 259\]](#).

Sample Code

```
"authTokens": [
  {
    "type": "Bearer",
    "value": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXLTJ5IiwiaWF0IjoiYm9keiJ9",
    "http_header": {
      "key": "Authorization",
      "value": "Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXLTJ5IiwiaWF0IjoiYm9keiJ9"
    }
  }
]
```

[Back to Content \[page 112\]](#)

Related Information

[Exchanging User JWTs via OAuth2UserTokenExchange Destinations \[page 263\]](#)

1.1.3.2.7 OAuth Password Authentication

Learn about the OAuth password authentication type for HTTP destinations in the Cloud Foundry environment: use cases, supported properties and examples.

Content

[Overview \[page 118\]](#)

[Properties \[page 118\]](#)

[Example: OAuth Token Service \[page 122\]](#)

Overview

SAP BTP provides support for applications to use the OAuth password grant flow for consuming OAuth-protected resources.

The client credentials as well as the user name and password are used to request an access token from an OAuth server, referred to as *token service* below. Access token retrieval is performed automatically by the Destination service when using the "find destination" REST endpoint.

Back to [Content \[page 118\]](#)

Properties

The table below lists the destination properties needed for the OAuth2Password authentication type.

⚠ Caution

Do not use your own *personal credentials* in the `<User>` and `<Password>` fields. Always use a *technical user* instead.

Property	Description
Required	
Name	Destination name. It must be the same as the destination name you use for the configuration tools, that is, the console client and <i>Destinations</i> editor (cockpit).

Property	Description
Type	Destination type. Choose HTTP (for HTTP or HTTPS communication).
URL	URL of the protected resource being accessed.
ProxyType	You can only use proxy type Internet or OnPremise . If OnPremise is used, the OAuth server must be accessed through the Cloud Connector.
Authentication	Authentication type. Use OAuth2Password as value.
clientId	Client ID used to retrieve the access token.
clientSecret	Client secret for the client ID.
User	User name of the technical user trying to get a token.
Password	Password of the technical user trying to get a token.
tokenServiceURL	Token retrieval URL of the OAuth server.
Additional	
scope	Value of the OAuth 2.0 <code>scope</code> parameter, expressed as a list of space-delimited, case-sensitive strings.
tokenServiceURL.headers.<header-key>	Static key prefix used as a namespace grouping of the <code>tokenServiceURL</code> 's HTTP headers. Its values will be sent to the token service during token retrieval. For each HTTP header's key you must add a <code>'tokenServiceURL.headers'</code> prefix separated by dot delimiter. For example:
<div style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>Sample Code</p> <pre> { ... "tokenServiceURL.headers.<header-key-1>" : "<header-value-1>", ... "tokenServiceURL.headers.<header-key-N>" : "<header-value-N>", } </pre> </div>	
tokenServiceURL.ConnectionTimeoutInSeconds	Defines the connection timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 60 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.
tokenServiceURL.SocketReadTimeoutInSeconds	Defines the read timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 600 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.

Property	Description
<code>tokenServiceURL.queries.<query-key></code>	<p>Static key prefix used as a namespace grouping of <code>tokenServiceURL</code>'s query parameters. Its values will be sent to the token service during token retrieval. For each query parameter's key you must add a <code>'tokenServiceURL.queries'</code> prefix separated by dot delimiter. For example:</p> <pre> ❏ Sample Code { ... "tokenServiceURL.queries.<query-key-1>" : "<query-value-1>", ... "tokenServiceURL.queries.<query-key-N>" : "<query-value-N>", } </pre>
<code>tokenService.body.<param-key></code>	<p>A static key prefix used as a namespace grouping of parameters which are sent as part of the token request to the token service during token retrieval. For each request, a <code>tokenService.body</code> prefix must be added to the parameter key, separated by dot-delimiter. For example:</p> <pre> ❏ Sample Code { ... "tokenService.body.<param-key-1>" : "<param-value-1>", ... "tokenService.body.<param-key-N>" : "<param-value-N>", } </pre>
<code>tokenService.KeyStoreLocation</code>	<p>Contains the name of the certificate configuration to be used. This property is required when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133].</p>
<code>tokenService.KeyStorePassword</code>	<p>Contains the password for the certificate configuration (if one is needed) when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133].</p>
<code>tokenService.addClientCredentialsInBody</code>	<p>Specifies whether the client credentials should be placed in the request body of the token request, rather than the <code>Authorization</code> header. Default is true.</p>
<code>clientAssertion.destinationName</code>	<p>Name of the destination that provides client assertions when using client assertion authentication mechanism. Must be on the same subaccount or service instance as this destination. This is used in case of automated client assertion fetching by the service.</p> <p>For more information, see Client Assertion with Automated Assertion Fetching by the Service [page 154].</p>

Property	Description
<code>URL.headers.<header-key></code>	Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a <code>URL.headers</code> prefix separated by dot-delimiter. For example:

Sample Code

```
{
  ...
  "URL.headers.<header-key-1>" : "<header-value-1>",
  ...
  "URL.headers.<header-key-N>" : "<header-value-N>",
}
```

Note

This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.

<code>URL.queries.<query-key></code>	Static key prefix used as a namespace grouping of URL's query parameters whose values will be sent to the target endpoint. For each query parameter's key, you must add a <code>URL.queries</code> prefix separated by dot-delimiter. For example:
--	--

Sample Code

```
{
  ...
  "URL.queries.<query-key-1>" : "<query-value-1>",
  ...
  "URL.queries.<query-key-N>" : "<query-value-N>",
}
```

Note

This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.

Note

When the OAuth server is called, the caller side trusts the server based on the trust settings of the destination. For more information, see [Server Certificate Authentication \[page 89\]](#).

Back to [Content \[page 118\]](#)

Example: OAuth Token Service

Sample Code

```
{
  "Name": "SapOAuthPassGrant",
  "Type": "HTTP",
  "URL": "https://myapp.cfapps.sap.hana.ondemand.com/mypath",
  "ProxyType": "Internet",
  "Authentication": "OAuth2Password",
  "clientId": "my-client-id",
  "clientSecret": "my-client-pass",
  "User": "my-username",
  "Password": "my-password",
  "tokenServiceURL": "https://authentication.sap.hana.ondemand.com/oauth/
token"
}
```

The response for "find destination" contains an `authTokens` object in the format given below. For more information on the fields in `authTokens`, see ["Find Destination" Response Structure \[page 259\]](#).

Sample Code

```
"authTokens": [
  {
    "type": "Bearer",
    "value": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXLTJ5In0...",
    "http_header": {
      "key": "Authorization",
      "value": "Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXLTJ5In0..."
    }
  }
]
```

Back to [Content \[page 118\]](#)

1.1.3.2.8 OAuth JWT Bearer Authentication

Learn about the OAuth JWT bearer authentication type for HTTP destinations in the Cloud Foundry environment: use cases, supported properties and examples.

Content

[Overview \[page 123\]](#)

[Properties \[page 123\]](#)

[Example: AuthTokens Object Response \[page 128\]](#)

Overview

To allow an application to call another application, passing the user context, and fetch resources, the caller application must pass an access token. In this authorization flow, the initial user token is passed to the OAuth server as input data. This process is performed automatically by the Destination service, which helps simplifying the application development: You only have to construct the right request to the target URL, by using the outcome (another access token) of the service-side automation.

Back to [Content \[page 123\]](#)

Properties

To configure a destination of this authentication type, you must specify all the required properties. You can do this via SAP BTP cockpit (see [Create HTTP Destinations \[page 63\]](#)), or using the [Destination Service REST API \[page 80\]](#). The following table shows the properties along with their semantics.

Field/Parameter (Cockpit)	JSON Key	Description
Required		
Authentication	Authentication	OAuth2JWTBearer in this case.
Client ID	clientId	OAuth 2.0 client ID to be used for the user access token exchange.
Client Secret	clientSecret	OAuth 2.0 client secret to be used for the user access token exchange.
Name	Name	Name of the destination. Must be unique for the destination level.
Proxy Type	ProxyType	You can only use proxy type Internet or OnPremise . If OnPremise is used, the OAuth server must be accessed through the Cloud Connector.

Field/Parameter (Cockpit)	JSON Key	Description
Token Service URL	tokenServiceURL	<p>The URL of the token service, against which the token exchange is performed. Depending on the <code>Token Service URL Type</code>, this property is interpreted in different ways during the automatic token retrieval:</p> <ul style="list-style-type: none"> For Dedicated, the token service URL is taken as is. For Common, the token service URL is searched for the tenant placeholder <code>{tenant}</code>. <code>{tenant}</code> is resolved as the subdomain of the subaccount on behalf of which the caller is performing the call. If the placeholder is not found, <code>{tenant}</code> is inserted as a subdomain of the token service URL. See Automated Access Token Retrieval [page 264] for information about how the tenant is determined. The subaccount subdomain is mandated during creation of the subaccount, see Create a Subaccount. <p>Examples of interpreting the token service URL for the token service URL type Common, if the call to the Destination service is on behalf of a subaccount subdomain with value <code>mytenant</code>:</p> <ul style="list-style-type: none"> <code>https://authentication.us10.hana.ondemand.com/oauth/token</code> → <code>https://mytenant.authentication.us10.hana.ondemand.com/oauth/token</code> <code>https://{tenant}.authentication.us10.hana.ondemand.com/oauth/token</code> → <code>https://mytenant.authentication.us10.hana.ondemand.com/oauth/token</code> <code>https://authentication.myauthserver.com/{tenant}/oauth/token</code> → <code>https://authentication.myauthserver.com/tenant/mytenant/oauth/token</code> <code>https://oauth.{tenant}.myauthserver.com/token</code> → <code>https://oauth.mytenant.myauthserver.com/token</code>
Token Service URL Type	tokenServiceURLType	<ul style="list-style-type: none"> Choose Dedicated if the token service URL serves only a single tenant. Choose Common if the token service URL serves multiple tenants.
Type	Type	Choose HTTP (for HTTP or HTTPS communication).
URL	URL	URL of the target endpoint.
Optional		
Description	Description	A human-readable description of the destination.
Additional		

Field/Parameter (Cockpit)	JSON Key	Description
	scope	The value of the OAuth 2.0 scope parameter, expressed as a list of space-delimited, case-sensitive strings.
	tokenServiceURL.headers.<header-key>	<p>A static key prefix used as a namespace grouping of the tokenServiceURL's HTTP headers. Its values will be sent to the token service during token retrieval. For each HTTP header's key you must add a 'tokenServiceURL.headers' prefix separated by dot delimiter. For example:</p> <div data-bbox="576 620 1398 943" data-label="Code-Block"> <pre> ❏ Sample Code { ... "tokenServiceURL.headers.<header-key-1>" : "<header-value-1>", ... "tokenServiceURL.headers.<header-key-N>" : "<header-value-N>", } </pre> </div>
	tokenServiceURL.ConnectionTimeoutInSeconds	Defines the connection timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 60 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.
	tokenServiceURL.SocketReadTimeoutInSeconds	Defines the read timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 600 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.
	tokenServiceURL.queries.<query-key>	<p>A static key prefix used as a namespace grouping of tokenServiceURL's query parameters. Its values will be sent to the token service during token retrieval. For each query parameter's key you must add a 'tokenServiceURL.queries' prefix separated by dot delimiter. For example:</p> <div data-bbox="576 1429 1398 1751" data-label="Code-Block"> <pre> ❏ Sample Code { ... "tokenServiceURL.queries.<query-key-1>" : "<query-value-1>", ... "tokenServiceURL.queries.<query-key-N>" : "<query-value-N>", } </pre> </div>

Field/Parameter (Cockpit)	JSON Key	Description
	<code>tokenService.body.<param-key></code>	A static key prefix used as a namespace grouping of parameters which are sent as part of the token request to the token service during token retrieval. For each request, a <code>tokenService.body</code> prefix must be added to the parameter key, separated by dot-delimiter. For example:
<div style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>Sample Code</p> <pre> { ... "tokenService.body.<param-key-1>" : "<param-value-1>", ... "tokenService.body.<param-key-N>" : "<param-value-N>", } </pre> </div>		
	<code>x_user_token.jwks</code>	Base64-encoded <i>JSON web key set</i> , containing the signing keys which are used to validate the JWT provided in the <i>X-User-Token</i> header. For more information, see JWK Set Format .
	<code>x_user_token.jwks_uri</code>	URI of the <i>JSON web key set</i> , containing the signing keys which are used to validate the JWT provided in the <i>X-User-Token</i> header.
<div style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>⚠ Restriction</p> <p>If the value is a private endpoint (for example, <i>localhost</i>), the Destination service will not be able to perform the verification of the X-User-Token header when using the "Find Destination" API.</p> <p>For more information, see OpenID Connect Discovery.</p> </div>		

Field/Parameter (Cockpit)	JSON Key	Description
	<code>URL.headers.<header-key></code>	A static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a <code>URL.headers</code> prefix separated by dot-delimiter. For example:
		<div data-bbox="596 510 813 546" data-label="Section-Header">Sample Code</div> <pre data-bbox="612 577 1329 786"> { ... "URL.headers.<header-key-1>" : "<header-value-1>", ... "URL.headers.<header-key-N>" : "<header-value-N>", }</pre>
		<div data-bbox="596 864 697 896" data-label="Section-Header">Note</div> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p>
	<code>URL.queries.<query-key></code>	A static key prefix used as a namespace grouping of URL's query parameters whose values will be sent to the target endpoint. For each query parameter's key, you must add a <code>URL.queries</code> prefix separated by dot-delimiter. For example:
		<div data-bbox="596 1249 813 1285" data-label="Section-Header">Sample Code</div> <pre data-bbox="612 1317 1355 1498"> { ... "URL.queries.<query-key-1>" : "<query-value-1>", ... "URL.queries.<query-key-N>" : "<query-value-N>", }</pre>
		<div data-bbox="596 1576 697 1608" data-label="Section-Header">Note</div> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p>
<code>tokenService.KeyStoreLocation</code>		Contains the name of the certificate configuration to be used. This property is required when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133] .

Field/Parameter (Cockpit)	JSON Key	Description
	tokenService.e.KeyStorePassword	Contains the password for the certificate configuration (if one is needed) when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133] .
	tokenService.addClientCredentialsInBody	Specifies whether the client credentials should be placed in the request body of the token request, rather than the Authorization header. Default is true .

Back to [Content \[page 123\]](#)

Example: AuthTokens Object Response

The response for "find destination" contains an `authTokens` object in the format given below. For more information on the fields in `authTokens`, see ["Find Destination" Response Structure \[page 259\]](#).

Sample Code

```

"authTokens": [
  {
    "type": "Bearer",
    "value": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXLTJ5IiwiaWF0IjoiMTYxODU2MTEwLjE1MzU0NjA1NzIifQ..",
    "http_header": {
      "key": "Authorization",
      "value": "Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXLTJ5IiwiaWF0IjoiMTYxODU2MTEwLjE1MzU0NjA1NzIifQ..",
    }
  }
]

```

Back to [Content \[page 123\]](#)

1.1.3.2.9 SAML Assertion Authentication

Create and configure an *SAML Assertion* destination for an application in the Cloud Foundry environment.

Context

The Destination service lets you generate SAML assertions as per SAML 2.0 specification. You can retrieve a generated SAML assertion from the Destination service by using the `SAMLAssertion` authentication type,

whereas [OAuth SAML Bearer Assertion Authentication \[page 94\]](#) sends the generated SAML assertion to an OAuth server to get a token. The Destination service provides functionality for caching the generated SAML assertion for later use, and caching by the app whenever needed, which helps simplifying application development.

Properties

The table below lists the destination properties for the *SAMLAssertion* authentication type.

Property	Description
Required	
Name	Name of the destination. Must be unique for the destination level.
Type	Destination type. Choose HTTP for all HTTP(S) destinations.
URL	URL of the target endpoint.
ProxyType	Choose Internet , PrivateLink , or OnPremise .
CloudConnectorLocationId	(only if ProxyType=OnPremise) Starting with Cloud Connector 2.9.0, you can connect multiple Cloud Connectors to an account , PrivateLinkas long as their location ID is different. The value defines the location ID identifying the Cloud Connector over which the connection is opened. The default value is an empty string identifying the Cloud Connector that is connected without any location ID, which is also the case for all Cloud Connector versions prior to 2.9.0.
Authentication	Authentication type. Use SAMLAssertion as value.
audience	Value of the Audience tag, which is part of the generated SAML assertion. For more information, see SAML 2.0 specification .
authnContextClassRef	Value of the AuthnContextClassRef tag, which is part of generated SAML assertion. For more information, see SAML 2.0 specification .
Additional	
clientKey	Key that identifies the consumer to the authorization server.
nameQualifier	When this property is set, the NameQualifier under the NameId tag of the generated SAML assertion is determined in accordance to the value.
companyId	Company identifier.
assertionIssuer	Issuer of the SAML assertion.
assertionRecipient	Recipient of the SAML assertion.

Property	Description
<code>nameIdFormat</code>	Value of the <code>NameIdFormat</code> tag, which is part of generated SAML Assertion. For more information, see SAML 2.0 specification .
<code>userIdSource</code>	When this property is set, the user ID in the <code>NameId</code> tag of the generated SAML assertion is determined in accordance to the value of this attribute. For more information, see User Propagation via SAML 2.0 Bearer Assertion Flow [page 251] .
<code>KeyStoreLocation</code>	Contains the name of the certificate configuration to be used for <i>per-destination</i> SAML assertion signing. This certificate will be used instead of the standard subaccount-wide signing key. For more information, see Set up Trust Between Systems [page 172] .
<code>KeyStorePassword</code>	Contains the password for the certificate configuration (if one is needed) when using a <i>per-destination</i> SAML assertion signing certificate.
<code>x_user_token.jwks</code>	Base64-encoded <i>JSON web key set</i> , containing the signing keys which are used to validate the JWT provided in the <i>X-User-Token</i> header. For more information, see JWK Set Format .
<code>x_user_token.jwks_uri</code>	URI of the <i>JSON web key set</i> , containing the signing keys which are used to validate the JWT provided in the <i>X-User-Token</i> header.

Restriction

If the value is a private endpoint (for example, *localhost*), the Destination service will not be able to perform the verification of the *X-User-Token* header when using the "Find Destination" API.

For more information, see [OpenID Connect Discovery](#) .

Property	Description
URL.headers.<header-key>	<p data-bbox="801 349 1399 495">A static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a URL.headers prefix separated by dot-delimiter. For example:</p> <div data-bbox="801 533 1399 853"><p data-bbox="826 544 1043 575">Sample Code</p><pre data-bbox="842 607 1383 819">{ ... "URL.headers.<header-key-1>" : "<header-value-1>", ... "URL.headers.<header-key-N>" : "<header-value-N>", }</pre></div> <div data-bbox="801 891 1399 1178"><p data-bbox="826 902 927 934">Note</p><p data-bbox="826 954 1358 1155">This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p></div>

Property	Description
<code>URL.queries.<query-key></code>	<p>A static key prefix used as a namespace grouping of URL's query parameters whose values will be sent to the target endpoint. For each query parameter's key, you must add a <code>URL.queries</code> prefix separated by dot-delimiter. For example:</p> <div data-bbox="826 539 1043 573" data-label="Section-Header"> <p>Sample Code</p> </div> <div data-bbox="842 607 1324 813" data-label="Code-Block"> <pre>{ ... "URL.queries.<query-key-1>" : "<query-value-1>", ... "URL.queries.<query-key-N>" : "<query-value-N>", }</pre> </div> <div data-bbox="826 893 927 925" data-label="Section-Header"> <p>Note</p> </div> <div data-bbox="826 949 1356 1151" data-label="Text"> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p> </div>
<code>skipUserAttributesPrefixInSAMLAttributes</code>	<p>If set to true, any additional attributes taken from the OAuth server's user information endpoint, under the <code>user_attributes</code> section, will be added to the assertion without the prefix that the Destination service would usually add to them. For more information, see User Propagation via SAML 2.0 Bearer Assertion Flow [page 251].</p>

Example

The connectivity destination below provides HTTP access to the OData API of the SuccessFactors Jam.

```
Name=destinationSamlAssertion
Type=HTTP
URL=https://myXXXXXX-api.s4hana.ondemand.com
Authentication=SAMLAssertion
ProxyType=Internet
audience=https://myXXXXXX.s4hana.ondemand.com
authnContextClassRef=urn:oasis:names:tc:SAML:2.0:ac:classes:X509
```

The response for "find destination" contains an `authTokens` object in the format given below. For more information on the fields in `authTokens`, see ["Find Destination" Response Structure \[page 259\]](#).

Sample Code

```
"authTokens": [
  {
    "type": "SAML2.0",
    "value": "PD94bWwgdmVyc2lvcj0iMS4wIiBlbmNvZ...",
    "http_header": {
      "key": "Authorization",
      "value": "SAML2.0 PD94bWwgdmVyc2lvcj0iMS4wIiBlbmNvZ..."
    }
  }
]
```

Related Information

[Create HTTP Destinations \[page 63\]](#)

[Destination Examples \[page 68\]](#)

[Exchanging User JWTs via OAuth2UserTokenExchange Destinations \[page 263\]](#)

1.1.3.2.10 OAuth with X.509 Client Certificates

Use an X.509 certificate instead of a secret to authenticate against the authentication server.

⚠ Caution

OAuth with X.509 is only supported for `<ProxyType>=Internet`.

To perform mutual TLS, you can use an X.509 client certificate instead of a client secret when connecting to the authorization server. To do so, you must create a certificate configuration containing a valid X.509 client certificate or a keystore, and link it to the destination configuration using these properties:

Property	Description
<code>tokenService.KeyStoreLocation</code>	Contains the name of the certificate configuration to be used. This property is required.
<code>tokenService.KeyStorePassword</code>	Contains the password for the certificate configuration (if one is needed).

⚠ Caution

Mutual TLS with an X.509 client certificate is performed only if the `tokenService.KeyStoreLocation` property is set in the destination configuration. Otherwise, the client secret is used.

Supported Certificate Configuration Formats

- Java Keystore (.jks): Requires the `tokenService.KeyStorePassword` property.
- PKCS12 (.pfx or .p12): Requires the `tokenService.KeyStorePassword` property.
- PEM-encoded X.509 client certificate and private key (.crt, .cer and .pem): The certificate configuration can contain several valid X.509 certificates and private keys.

Supported OAuth Flows

- [OAuth Client Credentials Authentication \[page 104\]](#)
- [OAuth Password Authentication \[page 118\]](#)
- [OAuth User Token Exchange Authentication \[page 112\]](#)
- [OAuth JWT Bearer Authentication \[page 122\]](#)
- [OAuth SAML Bearer Assertion Authentication \[page 94\]](#)

1.1.3.2.11 OAuth Refresh Token Authentication

Create and configure an *OAuth refresh token* destination for an application in the Cloud Foundry environment.

Overview

SAP BTP provides support for applications to use the *OAuth2 refresh token* flow for consuming OAuth-protected resources.

Refresh tokens are a common way to maintain certain levels of access, without requiring the use of credentials for getting a new access token. They have a longer validity compared to access tokens and can be used to fetch brand new access tokens without performing again the original flow.

The client credentials and a refresh token are used to request an access token from an OAuth server, referred to below as *token service*. This is automatically performed by the Destination service when using the "Find a destination" REST endpoint.

Properties

The table below lists the destination properties for the *OAuth refresh token* authentication type.

Property	Description
Required	
Name	Name of the destination. It must be the same as the destination name you use for the configuration tools, that is, the console client and <i>Destinations</i> editor (cockpit).
Type	Destination type. Choose HTTP for all HTTP(S) destinations.
URL	The URL of the protected target resource.
ProxyType	Choose Internet or OnPremise . If OnPremise is used, the OAuth server must be accessed through the Cloud Connector.
Authentication	Authentication type. Use OAuth2RefreshToken as value.
clientId	Client ID used to retrieve the access token.
clientSecret	Client secret for the Client ID.
tokenServiceURL	Token retrieval URL of the OAuth server.
tokenServiceURLType	Either Dedicated (if the <code>tokenServiceURL</code> serves only a single tenant), or Common (if the <code>tokenServiceURL</code> serves multiple tenants).
Additional	
scope	Value of the OAuth 2.0 <code>scope</code> parameter, expressed as a list of space-delimited, case-sensitive strings.
<code>tokenServiceURL.headers.<header-key></code>	Static key prefix used as a namespace grouping of the <code>tokenServiceURL</code> 's HTTP headers. Its values will be sent to the token service during token retrieval. For each HTTP header's key you must add a <code>'tokenServiceURL.headers'</code> prefix separated by dot delimiter. For example:

Sample Code

```
{
  ...
  "tokenServiceURL.headers.<header-key-1>" : "<header-value-1>",
  ...
  "tokenServiceURL.headers.<header-key-N>" : "<header-value-N>",
}
```

Property	Description
<code>tokenServiceURL.queries.<query-key></code>	<p>Static key prefix used as a namespace grouping of <code>tokenServiceURL</code>'s query parameters. Its values will be sent to the token service during token retrieval. For each query parameter's key you must add a <code>'tokenServiceURL.queries'</code> prefix separated by dot delimiter. For example:</p> <pre> { } Sample Code { ... "tokenServiceURL.queries.<query-key-1>" : "<query-value-1>", ... "tokenServiceURL.queries.<query-key-N>": "<query-value-N>", } </pre>
<code>tokenService.body.<param-key></code>	<p>A static key prefix used as a namespace grouping of parameters which are sent as part of the token request to the token service during token retrieval. For each request, a <code>tokenService.body</code> prefix must be added to the parameter key, separated by dot-delimiter. For example:</p> <pre> { } Sample Code { ... "tokenService.body.<param-key-1>" : "<param-value-1>", ... "tokenService.body.<param-key-N>": "<param-value-N>", } </pre>
<code>tokenService.KeyStoreLocation</code>	<p>Contains the name of the certificate configuration to be used. This property is required when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133].</p>
<code>tokenService.KeyStorePassword</code>	<p>Contains the password for the certificate configuration (if one is needed) when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133].</p>
<code>tokenService.addClientCredentialsInBody</code>	<p>Specifies whether the client credentials should be placed in the request body of the token request, rather than the <code>Authorization</code> header. Default is true.</p>

Property	Description
URL.headers.<header-key>	<p>Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a URL.headers prefix separated by dot-delimiter. For example:</p> <div data-bbox="826 539 1043 573" data-label="Section-Header"> <p>Sample Code</p> </div> <div data-bbox="842 607 1337 815" data-label="Code-Block"> <pre>{ ... "URL.headers.<header-key-1>" : "<header-value-1>", ... "URL.headers.<header-key-N>" : "<header-value-N>", }</pre> </div> <div data-bbox="826 898 927 931" data-label="Section-Header"> <p>Note</p> </div> <div data-bbox="826 954 1358 1151" data-label="Text"> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p> </div>
tokenServiceURL.ConnectionTimeoutInSeconds	<p>Defines the connection timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 60 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.</p>
tokenServiceURL.SocketReadTimeoutInSeconds	<p>Defines the read timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 600 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.</p>

Property	Description
<code>URL.queries.<query-key></code>	<p>Static key prefix used as a namespace grouping of URL's query parameters whose values will be sent to the target endpoint. For each query parameter's key, you must add a <code>URL.queries</code> prefix separated by dot-delimiter. For example:</p> <div data-bbox="826 539 1043 573" data-label="Section-Header"> <p>Sample Code</p> </div> <pre data-bbox="842 607 1326 815"> { ... "URL.queries.<query-key-1>" : "<query-value-1>", ... "URL.queries.<query-key-N>": "<query-value-N>", } </pre>

Note

Note

When the OAuth server is called, the caller side trusts the server based on the trust settings of the destination. For more information, see [Server Certificate Authentication \[page 89\]](#).

Example: OAuth2 RefreshToken Destination

Sample Code

```

{
  "Name": "SapOAuthPassGrant",
  "Type": "HTTP",
  "URL": "https://myapp.cfapps.sap.hana.ondemand.com/mypath",
  "ProxyType": "Internet",
  "Authentication": "OAuth2RefreshToken",
  "clientId": "my-client-id",
  "clientSecret": "my-client-pass",
  "tokenServiceURL": "https://authentication.sap.hana.ondemand.com/oauth/
token"
}

```

Calling "Find Destination"

When calling the destination an *X-refresh-token* is a required header parameter.

Curl call example

Sample Code

```
curl --location --request  
GET 'https://<destination>.<environment>.hanavlab.ondemand.com/destination-  
configuration/v1/destinations/<destination-name>' \  
--header 'X-refresh-token: <refresh_token>' \ # mandatory parameter  
--header 'Authorization: Bearer <destination_token>'
```

The response for *Find Destination* will contain an `authTokens` object in the format given below. For more information on the fields in `authTokens`, see ["Find Destination" Response Structure \[page 259\]](#).

Sample Code

```
"authTokens": [  
  {  
    "type": "Bearer",  
    "value": "eyJhbGciOiJSUzI1NiIsInR5cCI6IiJhbmR5bGci",  
    "http_header": {  
      "key": "Authorization",  
      "value": "Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IiJhbmR5bGci"  
    }  
  }  
]
```

1.1.3.2.12 OAuth Authorization Code Authentication

Create and configure an *OAuth Authorization Code* destination for an application in the Cloud Foundry environment.

Overview

The *OAuth Authorization Code* flow is a standard mechanism for business user login. It is a two-step procedure. In a first step, business users authenticate themselves towards an authorization server, which grants users an authorization code. The second step exchanges the authorization code for an access token through a token service. Applications can use this flow to access OAuth-protected resources.

The client credentials and an authorization code are used to request an access token from an OAuth server, referred to below as *token service*. This is performed automatically by the Destination service when using the "Find a destination" REST endpoint.

⚠ Restriction

This authentication type is not yet available for destination configuration via the cockpit.

Properties

The table below lists the destination properties for the *OAuth2AuthorizationCode* authentication type.

Property	Description
Required	
Name	Name of the destination. It must be the same as the destination name you use for the configuration tools, that is, the console client and <i>Destinations</i> editor (cockpit).
Type	Destination type. Choose HTTP for all HTTP(S) destinations.
URL	The URL of the protected target resource.
ProxyType	Choose Internet or OnPremise . If OnPremise is used, the OAuth server must be accessed through the Cloud Connector.
Authentication	Authentication type. Use OAuth2AuthorizationCode as value.
clientId	Client ID of the application.
clientSecret	Client secret for the Client ID.
tokenServiceURL	Token retrieval URL of the OAuth server.
tokenServiceURLType	Either Dedicated (if the tokenServiceURL serves only a single tenant), or Common (if the tokenServiceURL serves multiple tenants).
Additional	
scope	Value of the OAuth 2.0 <i>scope</i> parameter, expressed as a list of space-delimited, case-sensitive strings.

Property	Description
tokenServiceURL.headers.<header-key>	<p>Static key prefix used as a namespace grouping of the tokenServiceURL's HTTP headers. Its values will be sent to the token service during token retrieval. For each HTTP header's key you must add a 'tokenServiceURL.headers' prefix separated by dot delimiter. For example:</p> <div data-bbox="826 555 1043 589" data-label="Section-Header"> <p>↔ Sample Code</p> </div> <pre data-bbox="842 622 1326 880"> { ... "tokenServiceURL.headers.<header-key-1>" : "<header-value-1>", ... "tokenServiceURL.headers.<header-key-N>" : "<header-value-N>", } </pre>

Property	Description
<code>tokenService.body.<param-key></code>	<p>A static key prefix used as a namespace grouping of parameters which are sent as part of the token request to the token service during token retrieval. For each request, a <code>tokenService.body</code> prefix must be added to the parameter key, separated by dot-delimiter. For example:</p> <div data-bbox="804 533 1394 853" data-label="Code-Block"> <pre> ↔ Sample Code { ... "tokenService.body.<param- key-1>" : "<param-value-1>", ... "tokenService.body.<param-key- N>" : "<param-value-N>", } </pre> </div>
<code>tokenService.KeyStoreLocation</code>	<p>Contains the name of the certificate configuration to be used. This property is required when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133].</p>
<code>tokenService.KeyStorePassword</code>	<p>Contains the password for the certificate configuration (if one is needed) when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133].</p>
<code>tokenService.addClientCredentialsInBody</code>	<p>Specifies whether the client credentials should be placed in the request body of the token request, rather than the <code>Authorization</code> header. Default is true.</p>
<code>clientAssertion.destinationName</code>	<p>Name of the destination that provides client assertions when using client assertion authentication mechanism. Must be on the same subaccount or service instance as this destination. This is used in case of automated client assertion fetching by the service.</p> <p>For more information, see Client Assertion with Automated Assertion Fetching by the Service [page 154].</p>

Property	Description
URL.headers.<header-key>	<p>Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a URL.headers prefix separated by dot-delimiter. For example:</p>
	<p>Sample Code</p> <pre data-bbox="842 607 1342 819">{ ... "URL.headers.<header-key-1>" : "<header-value-1>", ... "URL.headers.<header-key-N>" : "<header-value-N>", }</pre>
	<p>Note</p> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p>

Property	Description
URL.queries.<query-key>	<p>Static key prefix used as a namespace grouping of URL's query parameters whose values will be sent to the target endpoint. For each query parameter's key, you must add a URL.queries prefix separated by dot-delimiter. For example:</p> <div data-bbox="826 539 1043 573" data-label="Section-Header"> <p>Sample Code</p> </div> <pre data-bbox="842 607 1326 815"> { ... "URL.queries.<query-key-1>" : "<query-value-1>", ... "URL.queries.<query-key-N>": "<query-value-N>", }</pre>

Note

Note

When the OAuth server is called, the caller side trusts the server based on the trust settings of the destination. For more information, see [Server Certificate Authentication \[page 89\]](#).

Example: OAuth2 Authorization Code Destination

Sample Code

```

{
  "Name": "SapOAuth2AuthorizationCodeDestination",
  "Type": "HTTP",
  "URL": "https://myapp.cfapps.sap.hana.ondemand.com/mypath",
  "ProxyType": "Internet",
  "Authentication": "OAuth2AuthorizationCode",
  "clientId": "my-client-id",
  "clientSecret": "my-client-pass",
  "tokenServiceURL": "https://authentication.sap.hana.ondemand.com/oauth/
token"
}
```

Calling "Find Destination"

When calling the destination, an *X-code* is a required header parameter. *X-redirect-uri* and *X-code-verifier* are optional header parameters. They depend on the call for the authorization code fetch. If a redirect URI was specified in that call, the same redirect URI must be used as value for the *X-redirect-uri* header. If a code challenge was presented in the authorization code fetch request, a code verifier must be given as value for the *X-code-verifier* header.

Curl call example

Sample Code

```
curl --location --request
GET 'https://<destination>.<environment>.hanavlab.ondemand.com/destination-
configuration/v1/destinations/<destination-name>' \
--header 'X-code: <authorization_code>' \ # mandatory parameter
--header 'X-redirect-uri: <redirect_uri>' \ # optional parameter
--header 'X-code-verifier: <code_verifier>' \ # optional parameter
--header 'Authorization: Bearer <destination_token>'
```

The response for *Find Destination* will contain an `authTokens` object in the format given below. For more information on the fields in `authTokens`, see ["Find Destination" Response Structure \[page 259\]](#).

Sample Code

```
"authTokens": [
  {
    "type": "Bearer",
    "value": "eyJhbGciOiJSUzI1NiIsInR5cCI6IiJ0eS...\"",
    "http_header": {
      "key": "Authorization",
      "value": "Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IiJ0eS...\""
    }
  }
]
```

1.1.3.2.13 OAuth Technical User Propagation Authentication

Learn about the *OAuth2TechnicalUserPropagation* authentication type for HTTP destinations in the Cloud Foundry environment: use cases, supported properties and examples.

Overview

SAP BTP supports the propagation of technical users from the cloud application towards on-premise systems. In the Destination service, an access token representing the technical user is retrieved, which can then be sent in a header to the Connectivity service. This is similar to *principal propagation*, but in this case, a technical user is propagated instead of a business user.

The retrieval of the access token performs the OAuth 2.0 client credentials flow, according to the token service configurations in the destination. The token service is called from the Internet, not from the [Cloud Connector](#).

Note

The retrieved access token is cached for the duration of its validity.

Restriction

This authentication type is not yet available for destination configuration via the cockpit.

Properties

The table below lists the destination properties for the *OAuth2TechnicalUserPropagation* authentication type.

Property	Description
Required	
Name	Name of the destination. It must be the same as the destination name you use for the configuration tools, that is, the console client and <i>Destinations</i> editor (cockpit).
Type	Destination type. Choose HTTP for all HTTP(S) destinations.
URL	The URL of the protected target resource.
ProxyType	You can only use proxy type OnPremise . → Remember The token service is not accessed through the Cloud Connector, but through the Internet.
Authentication	Authentication type. Use OAuth2TechnicalUserPropagation as value.
clientId	Client ID of the application.
clientSecret	Client secret for the Client ID.

Property	Description
tokenServiceURL	<p>The URL of the token service, against which the token exchange is performed. Depending on the <code>Token Service URL Type</code>, this property is interpreted in different ways during the automatic token retrieval:</p> <ul style="list-style-type: none"> For Dedicated, the token service URL is taken as is. For Common, the token service URL is searched for the tenant placeholder <code>{tenant}</code>. <code>{tenant}</code> is resolved as the subdomain of the subaccount on behalf of which the caller is performing the call. If the placeholder is not found, <code>{tenant}</code> is inserted as a subdomain of the token service URL. Consult the Destination Service REST API [page 80] to see how the subdomain of the subaccount is specified. The subaccount subdomain is mandated during creation of the subaccount, see Create a Subaccount. <p>Examples of interpreting the <code>tokenServiceURL</code> for <code>tokenServiceURLType=Common</code>, if the call to the Destination service is on behalf of a subaccount subdomain with value mytenant:</p> <ul style="list-style-type: none"> <code>https://authentication.us10.hana.ondemand.com/oauth/token</code> → <code>https://mytenant.authentication.us10.hana.ondemand.com/oauth/token</code> <code>https://{tenant}.authentication.us10.hana.ondemand.com/oauth/token</code> → <code>https://mytenant.authentication.us10.hana.ondemand.com/oauth/token</code> <code>https://authentication.myauthserver.com/tenant/{tenant}/oauth/token</code> → <code>https://authentication.myauthserver.com/tenant/mytenant/oauth/token</code> <code>https://oauth.{tenant}.myauthserver.com/token</code> → <code>https://oauth.mytenant.myauthserver.com/token</code>
tokenServiceURLType	<p>Either Dedicated (if the <code>tokenServiceURL</code> serves only a single tenant), or Common (if the <code>tokenServiceURL</code> serves multiple tenants).</p>

Property	Description
<code>tokenServiceUser</code>	User for basic authentication to the OAuth server (if required).
<code>tokenServicePassword</code>	Password for <code>tokenServiceUser</code> (if required).
Additional	
<code>scope</code>	Value of the OAuth 2.0 <code>scope</code> parameter, expressed as a list of space-delimited, case-sensitive strings.
<code>tokenServiceURL.headers.<header-key></code>	Static key prefix used as a namespace grouping of the <code>tokenServiceURL</code> 's HTTP headers. Its values will be sent to the token service during token retrieval. For each HTTP header's key you must add a <code>'tokenServiceURL.headers'</code> prefix separated by dot delimiter. For example: <div data-bbox="804 801 1394 1173" data-label="Code-Block"> <pre> ↳ Sample Code { ... "tokenServiceURL.headers.<header-key-1>" : "<header-value-1>", ... "tokenServiceURL.headers.<header-key-N>" : "<header-value-N>", } </pre> </div>
<code>tokenServiceURL.ConnectionTimeoutInSeconds</code>	Defines the connection timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 60 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.
<code>tokenServiceURL.SocketReadTimeoutInSeconds</code>	Defines the read timeout for the token service retrieval. The minimum value allowed is 0, the maximum is 600 seconds. If the value exceeds the allowed number, the default value (10 seconds) is used.

Property	Description
<code>tokenServiceURL.queries.<query-key></code>	<p>Static key prefix used as a namespace grouping of <code>tokenServiceUrl</code>'s query parameters. Its values will be sent to the token service during token retrieval. For each query parameter's key you must add a <code>'tokenServiceURL.queries'</code> prefix separated by dot delimiter. For example:</p> <div data-bbox="826 584 1396 952" style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>Sample Code</p> <pre>{ ... "tokenServiceURL.queries.<query-key-1>" : "<query-value-1>", ... "tokenServiceURL.queries.<query-key-N>": "<query-value-N>", }</pre> </div>
<code>tokenService.body.<param-key></code>	<p>A static key prefix used as a namespace grouping of parameters which are sent as part of the token request to the token service during token retrieval. For each request, a <code>tokenService.body</code> prefix must be added to the parameter key, separated by dot-delimiter. For example:</p> <div data-bbox="826 1160 1396 1480" style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>Sample Code</p> <pre>{ ... "tokenService.body.<param-key-1>" : "<param-value-1>", ... "tokenService.body.<param-key-N>": "<param-value-N>", }</pre> </div>
<code>tokenService.KeyStoreLocation</code>	<p>Contains the name of the certificate configuration to be used. This property is required when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133].</p>
<code>tokenService.KeyStorePassword</code>	<p>Contains the password for the certificate configuration (if one is needed) when using client certificates for authentication. See OAuth with X.509 Client Certificates [page 133].</p>

Property	Description
<code>tokenService.addClientCredentialsInBody</code>	<p>Specifies whether the client credentials should be placed in the request body of the token request, rather than the <code>Authorization</code> header. Default is <code>true</code>.</p> <div data-bbox="821 459 1394 683" style="border: 1px solid #ccc; padding: 5px;"> <p>Note</p> <p>If set to <code>false</code>, but <code>tokenServiceUser / tokenServicePassword</code> are also set, <code>tokenServiceUser / tokenServicePassword</code> will be taken with priority.</p> </div>
<code>clientAssertion.destinationName</code>	<p>Name of the destination that provides client assertions when using client assertion authentication mechanism. Must be on the same subaccount or service instance as this destination. This is used in case of automated client assertion fetching by the service.</p> <p>For more information, see Client Assertion with Automated Assertion Fetching by the Service [page 154].</p>
<code>URL.headers.<header-key></code>	<p>Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a <code>URL.headers</code> prefix separated by dot-delimiter. For example:</p> <div data-bbox="821 1142 1394 1467" style="border: 1px solid #ccc; padding: 5px;"> <p>Sample Code</p> <pre>{ ... "URL.headers.<header-key-1>" : "<header-value-1>", ... "URL.headers.<header-key-N>" : "<header-value-N>", }</pre> </div> <div data-bbox="821 1500 1394 1792" style="border: 1px solid #ccc; padding: 5px;"> <p>Note</p> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p> </div>

Property	Description
<code>URL.queries.<query-key></code>	<p>Static key prefix used as a namespace grouping of URL's query parameters whose values will be sent to the target endpoint. For each query parameter's key, you must add a <code>URL.queries</code> prefix separated by dot-delimiter. For example:</p> <pre> ❏ Sample Code { ... "URL.queries.<query-key-1>" : "<query-value-1>", ... "URL.queries.<query-key-N>" : "<query-value-N>", }</pre>
	<p>❏ Note</p> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention.</p>

❏ Note

When the OAuth authorization server is called, it accepts the trust settings of the destination. For more information, see [Server Certificate Authentication \[page 89\]](#).

⚠ Caution

When using the OAuth service of the SAP BTP Neo environment (`https://api.{landscape-domain}/oauth2/apitoken/v1?grant_type=client_credentials` or `oauthservices.{landscape-domain}/oauth2/apitoken/v1?grant_type=client_credentials`) as `TokenServiceURL`, or any other OAuth token service which accepts client credentials **only as Authorization header**, you must also set the `clientId` and `clientSecret` values for `tokenServiceUser` and `tokenServicePassword` properties.

Example: OAuth Technical User Propagation Destination

❏ Sample Code

```
Name=technical-user-example
```

```
Type=HTTP
URL=http://protected-url.example.com
ProxyType=OnPremise
Authentication=OAuth2TechnicalUserPropagation
clientId=clientId
clientSecret=secret1234
tokenServiceURL=http://authserver.example.com/oauth/token
```

Example: authTokens Object in *Find Destination* Response

The response for *Find Destination* will contain an `authTokens` object in the format given below. For more information on the fields in `authTokens`, see ["Find Destination" Response Structure \[page 259\]](#).

For usage of the `SAP-Connectivity-Technical-Authentication` header, see [Authentication Types \[page 224\]](#).

Sample Code

```
"authTokens": [
  {
    "type": "Bearer",
    "value": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiI... ",
    "http_header": {
      "key": "SAP-Connectivity-Technical-Authentication",
      "value": "Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiI..."
    }
  }
]
```

1.1.3.2.14 Using Client Assertion with OAuth Flows

Replace client secrets with client assertions in OAuth flows for destinations in the Cloud Foundry environment.

Client assertion is one of the possible client authentication mechanisms when requesting an access token from an authorization server. When using client assertion, you must provide the `client_assertion` and `client_assertion_type` parameters in the request to an authorization server that supports client assertion for client authentication.

The Destination service supports client assertion as a client authentication mechanism. It sends the client assertion to the authorization server whenever requesting an OAuth token. The assertion can be either generated externally and provided to the Destination service, or it can be retrieved by the Destination service via an additional destination.

For more information, see [Provide Client Assertion Properties as Headers \[page 153\]](#) and [Client Assertion with Automated Assertion Fetching by the Service \[page 154\]](#).

Prerequisites

You must ensure that the target authorization server supports client assertion as authentication mechanism and accepts the provided client assertions after validation. This might require additional setup in the authorization server and the assertion provider.

1.1.3.2.14.1 Provide Client Assertion Properties as Headers

Provide client assertion properties as headers when using client assertion with OAuth flows for a destination.

You can provide the following headers to use client assertion authentication:

Header	Value	Description
X-client-assertion-type	Absolute URI	Format of the assertion as defined by the authorization server. The value is an absolute URI. A URN of the form <i>urn:ietf:params:oauth:client-assertion-type:*</i> is suggested. Examples: <ul style="list-style-type: none">"urn:ietf:params:oauth:client-assertion-type:saml2-bearer" => indicating a SAML Bearer assertion."urn:ietf:params:oauth:client-assertion-type:jwt-bearer" => indicating a JWT Bearer token.
X-client-assertion	Token	Assertion being used to authenticate the client.

Example: Destination for an OAuth Service Accepting Client Assertion instead of Client Secret

Sample Code

```
Name=sap_Destination
Type=HTTP
URL= https://xxxx.example.com
ProxyType=Internet
Authentication=OAuth2ClientCredentials
clientId=clientId
tokenServiceURL=https://authserver.example.com/oauth/token/
```


Example: Client Assertion Properties as Headers

To use the client assertion mechanism in the *Find Destination* API, you must add two mandatory headers as shown in the following example:

Sample Code

```
curl --location --request GET 'https://<destination_service_host>/destination-configuration/v1/destinations/<destination-name>' \
--header 'X-client-assertion-type: urn:ietf:params:oauth:client-assertion-type:jwt-bearer' \ # mandatory parameter
--header 'X-client-assertion: eyJraWQiOiJKMwpc...' \ # mandatory parameter
--header 'Authorization: Bearer <access_token>'
```

1.1.3.2.14.2 Client Assertion with Automated Assertion Fetching by the Service

Use the *Find Destination* API to fetch client assertions automatically when using client assertion with OAuth flows for a destination.

The *Find Destination* API lets you fetch client assertions automatically and then use them for retrieving tokens from authorization servers that accept client assertions as a client authentication mechanism.

To apply this mechanism, you must use the following of destination configurations:

- Destination that provides the client assertion - with specified token service that issues client assertions
- Destination that uses the client assertion - with specified token service that uses client assertion as a client authentication mechanism

Caution

Only *OAuth2ClientCredentials* authentication type is allowed for destinations that provide client assertions. The following authentication types are supported for destinations that use client assertions: *OAuth2Password*, *OAuth2ClientCredentials*, *OAuth2AuthorizationCode*, *OAuth2TechnicalUserPropagation*.

Define the Client Assertion Type

The client assertion type must be defined as a property in the destination that provides the client assertion:

Property	Value	Description
<code>clientAssertion.type</code>	Absolute URI	The format of the assertion as defined by the authorization server. The supported values are: <ul style="list-style-type: none"> "urn:ietf:params:oauth:client-assertion-type:saml2-bearer" => indicating a SAML Bearer Assertion. "urn:ietf:params:oauth:client-assertion-type:jwt-bearer" => indicating a JWT Bearer Token.

Use a Property to Add a Reference to the Destination that Provides the Client Assertion

The destination that provides the client assertion can be specified in a property of the destination that uses client assertions:

Property	Value	Description
<code>clientAssertion.destinationName</code>	Name of a destination	Name of the destination that provides the client assertion. Must be on the same subaccount or service instance as the destination that uses client assertions.

⚠ Caution

If headers `x-client-assertion` and `x-client-assertion-type` are specified in the *Find Destination* API call, the `clientAssertion.destinationName` property will not be used for an automated assertion fetching mechanism.

Use a Header to Specify the Destination that Provides the Client Assertion

Alternatively, the destination that provides the client assertion can also be specified in a header in the *Find Destination* API.

Header	Value	Description
X-client-assertion-destination-name	Name of a destination	Name of the destination that provides the client assertion. Must be on the same subaccount or service instance as the destination that uses client assertions.

If specified, this header overrides the property `clientAssertion.destinationName` in the destination that uses client assertions.

⚠ Caution

Usage of headers `X-client-assertion` and `X-client-assertion-type` is not allowed if this header is present.

Example: Destination that Provides Client Assertions

📄 Sample Code

```
Name=Provides_Client_Assertion_Destination
Type=HTTP
URL= https://xxxx.example.com
ProxyType=Internet
Authentication=OAuth2ClientCredentials
clientId=clientId
clientSecret=secret1234
tokenServiceURL=https://authserver1.example.com/oauth/token/
clientAssertion.type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
```

Example: Destination that Uses Client Assertions

📄 Sample Code

```
Name=Uses_Client_Assertion_Destination
Type=HTTP
URL= https://xxxx.example.com
ProxyType=Internet
Authentication=OAuth2ClientCredentials
clientId=clientId
tokenServiceURL=https://authserver2.example.com/oauth/token/
clientAssertion.destinationName=Provides_Client_Assertion_Destination
```

Example: Find Destination API Call for Client Assertion Authentication with Automated Assertion Fetching

curl call:

Sample Code

```
curl --location --request
GET 'https://<destination_service_host>/destination-configuration/v1/
destinations/<name_of_destination_that_uses_client_assertions>' \
--header 'Authorization: Bearer <access_token>'
```

1.1.3.3 RFC Destinations

RFC destinations provide the configuration required for communication with an on-premise ABAP system via Remote Function Call. The RFC destination data is used by the Java Connector (JCo) version that is available within SAP BTP to establish and manage the connection.

RFC Destination Properties

The RFC destination specific configuration in SAP BTP consists of properties arranged in groups, as described below. The supported set of properties is a subset of the standard JCo properties in arbitrary environments. The configuration data is divided into the following groups:

- [User Logon Properties \[page 158\]](#)
- [Pooling Configuration \[page 160\]](#)
- [Repository Configuration \[page 162\]](#)
- [Target System Configuration \[page 163\]](#)
- [Parameters Influencing Communication Behavior \[page 167\]](#)

The minimal configuration contains user logon properties and information identifying the target host. This means you must provide at least a set of properties containing this information.

Example

```
Name=SalesSystem
Type=RFC
jco.client.client=000
jco.client.lang=EN
jco.client.user=consultant
jco.client.passwd=<password>
```

```
jco.client.ashost=sales-system.cloud
jco.client.sysnr=42
jco.destination.pool_capacity=5
jco.destination.peak_limit=10
```

Related Information

[Invoking ABAP Function Modules via RFC \[page 277\]](#)

1.1.3.3.1 User Logon Properties

JCo properties that cover different types of user credentials, as well as the ABAP system client and the logon language.

The currently supported logon mechanism uses user or password as credentials.

Property	Description
<code>jco.client.client</code>	Represents the client to be used in the ABAP system. Valid format is a three-digit number.
<code>jco.client.lang</code>	Optional property. Represents the logon language. If the property is not provided, the user's or system's default language is used. Valid values are two-character ISO language codes or one-character SAP language codes.
<code>jco.client.user</code>	Represents the user to be used for logging on to the ABAP system. Max. 12 characters long. Note When working with the <i>Destinations</i> editor in the cockpit, enter the value in the <code><User></code> field. Do not enter it as additional property.
<code>jco.client.alias_user</code>	Represents the user to be used for logging on to the ABAP system. Either <code>jco.client.user</code> or <code>jco.client.alias_user</code> must be specified. The alias user may be up to 40 characters long. Note When working with the <i>Destinations</i> editor in the cockpit, enter the value in the <code><Alias User></code> field. Do not enter it as additional property.

Property	Description
jco.client.passwd	Represents the password of the user that is used. <div data-bbox="826 398 1382 600" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note</p> <p>Passwords in systems of SAP NetWeaver releases lower than 7.0 are case-insensitive and can be only eight characters long. For releases 7.0 and higher, passwords are case-sensitive with a maximum length of 40.</p> </div> <div data-bbox="826 627 1382 784" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note</p> <p>When working with the <i>Destinations</i> editor in the cockpit, enter this password in the <code><Password></code> field. Do not enter it as additional property.</p> </div>
jco.client.tls_client_certificate_logon	<p>When set to 1, the client certificate provided by the <i>KeyStore</i>, which must be configured in addition, is used for authentication instead of <code>jco.client.user/jco.client.alias_user</code> and <code>jco.client.passwd</code>. This property is only relevant for a connection using WebSocket RFC (<code><Proxy Type>=Internet</code>).</p> <p>The default value is 0.</p> <div data-bbox="826 1146 1382 1339" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note</p> <p>When working with the Destinations editor in the cockpit, the <code><User></code>, <code><Alias User></code> and <code><Password></code> fields are hidden when setting the property to 1.</p> </div> <p>For more information on WebSocket RFC, see also: WebSocket RFC</p>

Property	Description
<code>jco.destination.auth_type</code>	<p>Optional property.</p> <ul style="list-style-type: none"> If the property is not provided, its default value CONFIGURED_USER is used, which means that user, password, or other credentials are specified directly. To enable single sign-on via principal propagation (which means that the identity logged on in the cloud application is forwarded to the on-premise system), set the value to PrincipalPropagation. In this case, you do not need to provide <code>jco.client.user</code> and <code>jco.client.passwd</code> in the configuration. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Note</p> <p>For PrincipalPropagation, you should configure the properties <code>jco.destination.repository.user</code> and <code>jco.destination.repository.passwd</code> instead, since there are special permissions needed (for metadata lookup in the back end) that not all business application users might have.</p> </div>

1.1.3.3.2 Pooling Configuration

Learn about the JCo properties you can use to configure pooling in an RFC destination.

Overview

This group of JCo properties covers different settings for the behavior of the destination's connection pool. All properties are optional.

Property	Description
<code>jco.destination.pool_capacity</code>	Represents the maximum number of idle connections kept open by the destination. A value of 0 has the effect of no connection pooling, that is, connections will be closed after each request. The default value is 1 .

Property	Description
<code>jco.destination.peak_limit</code>	<p>Represents the maximum number of active connections you can create for a destination simultaneously. Value 0 allows an unlimited number of active connections. Otherwise, if the value is less than the value of <code>jco.destination.pool_capacity</code>, it will be automatically increased to this value.</p> <p>Default setting is the value of <code>jco.destination.pool_capacity</code>. If <code>jco.destination.pool_capacity</code> is not specified, the default is 0 (unlimited).</p>
<code>jco.destination.max_get_client_time</code>	<p>Represents the maximum time in milliseconds to wait for a free connection in case the maximum number of active connections is already allocated by applications. The default value is 30000 (30 seconds).</p>
<code>jco.destination.expiration_time</code>	<p>Represents the time in milliseconds after which idle connections that are available in the pool can be closed. The default value is 60000 (60 seconds).</p>
<code>jco.destination.expiration_check_period</code>	<p>Represents the interval in milliseconds for the timeout checker thread to check the idle connections in the pool for expiration. The default value is 60000 (60 seconds).</p>
<code>jco.destination.pool_check_connection</code>	<p>When setting this value to 1, a pooled connection will be checked for corruption before being used for the next function module execution. Thus, it is possible to recognize corrupted connections and avoid exceptions being passed to applications when connectivity is basically working (default value is 0).</p>

Note

Turning on this check has performance impact for stateless communication. This is due to an additional low-level ping to the server, which takes a certain amount of time for non-corrupted connections, depending on latency.

Pooling Details

- Each destination is associated with a connection factory and, if the pooling feature is used, with a connection pool.
- Initially, the destination's connection pool is empty, and the JCo runtime does not preallocate any connection. The first connection will be created when the first function module invocation is performed. The `peak_limit` property describes how many connections can be created simultaneously, if applications allocate connections in different sessions at the same time. A connection is allocated either when a

stateless function call is executed, or when a connection for a stateful call sequence is reserved within a session.

- After the `<peak_limit>` number of connections has been allocated (in `<peak_limit>` number of sessions), the next session will wait for at most `<max_get_client_time>` milliseconds until a different session releases a connection (either finishes a stateless call or ends a stateful call sequence). In case the waiting session does not get any connection during the `<max_get_client_time>` period, the function request will be aborted with `JCoException` with the key `JCO_ERROR_RESOURCE`.
- Connections that are no longer used by applications are returned to the destination pool. There is at most a `<pool_capacity>` number of connections kept open by the pool. Further connections (`<peak_limit>` - `<pool_capacity>`) will be closed immediately after usage. The pooled connections (open connections in the pool) are marked as expired if they are not used again during `<expiration_time>` milliseconds. All expired connections will be closed by a timeout checker thread which executes the check every `<expiration_check_period>` milliseconds.

1.1.3.3.3 Repository Configuration

JCo properties that allow you to define the behavior of the repository that dynamically retrieves function module metadata.

All properties below are optional. Alternatively, you can create the metadata in the application code, using the metadata factory methods within the `JCo` class, to avoid additional round-trips to the on-premise system.

Property	Description
<code>jco.destination.repository_destination</code>	Specifies which destination should be used for repository queries. If the destination does not exist, an error occurs when trying to retrieve the repository. Defaults to itself.
<code>jco.destination.repository.user</code>	Optional property. If this property is set, and the repository destination is not set, it is used as the user for repository queries. This configuration option allows using a different user for repository lookups with a single destination configuration, and restricting this user's permissions accordingly. See also SAP Note 460089 .
	<div style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 5px;"> <p>Note</p> <p>When working with the <i>Destinations</i> editor in the cockpit, enter the value in the <code><Repository User></code> field. Do not enter it as additional property.</p> </div>
<code>jco.destination.repository.passwd</code>	Represents the password for a repository user. If you use such a user, this property is mandatory.
	<div style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 5px;"> <p>Note</p> <p>When working with the <i>Destinations</i> editor in the cockpit, enter this password in the <code><Repository Password></code> field. Do not enter it as additional property.</p> </div>

1.1.3.3.4 Target System Configuration

Learn about the JCo properties you can use to configure the target system information in an RFC destination (Cloud Foundry environment).

Note

This documentation refers to SAP BTP, Cloud Foundry environment. If you are looking for information about the Neo environment, see [Target System Configuration](#) (Neo environment).

Content

[Overview \[page 163\]](#)

[Proxy Types \[page 164\]](#)

[Direct Connection \[page 164\]](#)

[Load Balancing Connection \[page 164\]](#)

[WebSocket Connection \[page 165\]](#)

Overview

You can use the following configuration types alternatively:

- Direct connection to an ABAP application server
- Load balancing connection to a group of ABAP application servers via a message server
- WebSocket connection to an ABAP application server (RFC over Internet)

Note

When using a WebSocket connection, the target ABAP system must be exposed to the Internet.

Depending on the configuration you use, different properties are mandatory or optional.

To improve performance, consider using optional properties additionally, such as `jco.client.serialization_format`. For more information, see [JCo documentation](#).

Back to [Content \[page 163\]](#)

Proxy Types

The field `<Proxy Type>` lets you choose between `Internet` and `OnPremise`. When choosing `OnPremise`, the RFC communication is routed over a Cloud Connector that is connected to the subaccount. When choosing `Internet`, the RFC communication is done over a WebSocket connection.

Back to [Content \[page 163\]](#)

Direct Connection

To use a direct connection (connection without load balancing) to an application server over Cloud Connector, you must set the value for `<Proxy Type>` to `OnPremise`.

Property	Description
<code>jco.client.ashost</code>	Represents the application server host to be used. For configurations on SAP BTP, the property must match a virtual host entry in the Cloud Connector <i>Access Control</i> configuration. The property indicates that a direct connection is established.
<code>jco.client.sysnr</code>	Represents the so-called "system number" and has two digits. It identifies the logical port on which the application server is listening for incoming requests. For configurations on SAP BTP, the property must match a virtual port entry in the Cloud Connector <i>Access Control</i> configuration. <div data-bbox="821 1323 1401 1491"><p>Note</p><p>The virtual port in the above access control entry must be named <code>sapgw<##></code>, where <code><##></code> is the value of <code>sysnr</code>.</p></div>
<code>jco.client.client</code>	Three-digit ABAP client number. Defines the client of the target ABAP system.

Back to [Content \[page 163\]](#)

Load Balancing Connection

To use load balancing to a system over Cloud Connector, you must set the value for `<Proxy Type>` to `OnPremise`.

Property	Description
<code>jco.client.mshost</code>	Represents the message server host to be used. For configurations on SAP BTP, the property must match a virtual host entry in the Cloud Connector <i>Access Control</i> configuration. The property indicates that load balancing is used for establishing a connection.
<code>jco.client.group</code>	Optional property. Identifies the group of application servers that is used, the so-called "logon group". If the property is not specified, the group PUBLIC is used.
<code>jco.client.r3name</code>	Represents the three-character system ID of the ABAP system to be addressed. For configurations on SAP BTP, the property must match a virtual port entry in the Cloud Connector <i>Access Control</i> configuration.
	<div style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 10px;"> <p>Note</p> <p>The virtual port in the above access control entry must be named <code>sapms<###></code>, where <code><###></code> is the value of <code>r3name</code>.</p> </div>
<code>jco.client.msserv</code>	Represents the port on which the message server is listening for incoming requests. you can use this property as an alternative to <code>jco.client.r3name</code> . One of these two must be present. For configurations on SAP BTP, the property must match a virtual port entry in the Cloud Connector <i>Access Control</i> configuration. You can therefore avoid look-ups in the <code>/etc/services</code> file (<code><Install_Drive>\Windows\System32\drivers\etc\services</code>) on the Cloud Connector host.
<code>jco.client.client</code>	Three-digit ABAP client number. Defines the client of the target ABAP system.

Back to [Content \[page 163\]](#)

WebSocket Connection

To use a direct connection over WebSocket, you must set the value for `<Proxy Type>` to Internet.

Prerequisites

- Your target system is an ABAP server as of S/4HANA (on-premise) version 1909, or a cloud ABAP system.
- Your SAP Java buildpack version is at least 1.26.0.

Property	Description
<code>jco.client.wshost</code>	Represents the WebSocket RFC server host on which the target ABAP system is running. The system must be exposed to the Internet.
<code>jco.client.wsport</code>	Represents the WebSocket RFC server port on which the target ABAP system is listening.
<code>jco.client.client</code>	Three-digit ABAP client number. Defines the client of the target ABAP system.
<code>jco.client.tls_trust_all</code>	If set to 1, all server certificates are considered trusted during TLS handshake. If set to 0, either a dedicated trust store must be configured, or the JDK trust store is used as default. Default value is 0.

Note

We recommend that you **do not use value 1 ("trust all") in productive scenarios**, but only for demo/test purposes.

<Trust Store Location>

1. When used in local environment
2. When used in cloud environment

If you don't want to use the default JDK trust store (option *Use default JDK truststore* is unchecked), you must enter a `<Trust Store Location>`. This field indicates the path to the JKS file which contains trusted certificates (Certificate Authorities) for authentication against a remote client.

1. The relative path to the JKS file. The root path is the server's location on the file system.
2. The name of the JKS file.

Note

If the `<Trust Store Location>` is not specified, the JDK trust store is used as a default trust store for the destination.

<Trust Store Password>

Password for the JKS trust store file. This field is mandatory if `<Trust Store Location>` is used.

Note

You can upload trust store JKS files using the same command as for uploading destination configuration property files. You only need to specify the JKS file instead of the destination configuration file.

Note

Connections to remote services which require *Java Cryptography Extension (JCE) unlimited strength jurisdiction policy* are not supported.

See also [WebSocket RFC](#) (ABAP Platform documentation).

1.1.3.3.5 Parameters Influencing Communication Behavior

JCo properties that allow you to control the connection to an ABAP system.

All properties are optional.

Property	Description
<code>jco.client.trace</code>	Defines whether protocol traces are created. Valid values are 1 (trace is on) and 0 (trace is off). The default value is 0 .
<code>jco.client.codepage</code>	Declares the 4-digit SAP codepage that is used when initiating the connection to the backend. The default value is 1100 (comparable to iso-8859-1). It is important to provide this property if the password that is used contains characters that cannot be represented in 1100 .
<code>jco.client.delta</code>	Enables or disables table parameter delta management. It is enabled if set to 1 , and respectively disabled if set to 0 . The default value is 1 .
<code>jco.client.cloud_connector_version</code>	Defines the Cloud Connector version used for establishing a connection to the on-premise system. The default value is 2 . Currently, no other values are supported.
<code>jco.client.cloud_connector_location_id</code>	You can connect multiple Cloud Connectors to a subaccount as long as their location ID is different. The location ID specifies the Cloud Connector over which the connection is opened. The default value is an empty string identifying the Cloud Connector that is connected without any location ID. <div data-bbox="821 1444 1396 1668"><p>Note</p><p>When working with the <i>Destinations</i> editor in the cockpit, enter the Cloud Connector location ID in the <code><Location ID></code> field. Do not enter it as additional property.</p></div>
<code>jco.client.serialization_format</code>	Defines the serialization format that is used when transferring function module data to the partner system. The property impacts the serialization behavior of function module data. Valid values are columnBased and rowBased . If you choose columnBased , the <i>fast RFC serialization</i> is used, as long as the partner system supports it, see SAP Note 2372888 . When choosing the rowBased option, <i>classic</i> or <i>basXML</i> serialization are used. The default value is rowBased .

Property	Description
<code>jco.client.network</code>	Defines which network type is expected to be used for the destination. The property impacts the serialization behavior of function module data, see SAP Note 2372888 . Valid values are WAN and LAN . The default value is LAN .

1.1.3.4 Principal Propagation

Enable single sign-on (SSO) by forwarding the identity of cloud users to a remote system or service (Cloud Foundry environment).

The Connectivity and Destination services let you forward the identity of a cloud user to a remote system. This process is called principal propagation (also known as *user propagation* or *user principal propagation*). It uses a JSON Web token (JWT) as exchange format for the user information.

Two scenarios are supported: Cloud to on-premise (using the Connectivity service) and cloud to cloud (using the Destination service).

- [Scenario: Cloud to On-Premise \[page 168\]](#): The user is propagated from a cloud application to an on-premise system using a destination configuration with authentication type `PrincipalPropagation`.

Note

This scenario requires the Cloud Connector to connect to your on-premise system.

- [Scenario: Cloud to Cloud \[page 170\]](#): The user is propagated from a cloud application to another remote (cloud) system using a destination configuration with authentication type `OAuth2SAMLBearerAssertion`.

For more information on setting up destinations, see:

- [Create HTTP Destinations \[page 63\]](#)
- [Create RFC Destinations \[page 65\]](#)

1.1.3.4.1 Scenario: Cloud to On-Premise

Forward the identity of cloud users from the Cloud Foundry environment to on-premise systems using principal propagation.

Concept

The Connectivity service lets you connect your cloud applications to on-premise systems through the Cloud Connector and forward the identity of a cloud user. This process is called principal propagation (also known as user propagation). The JSON Web token (JWT) representing the cloud user identity is forwarded to the Cloud Connector, which verifies it, and propagates the user identity via either an X.509 certificate or Kerberos.

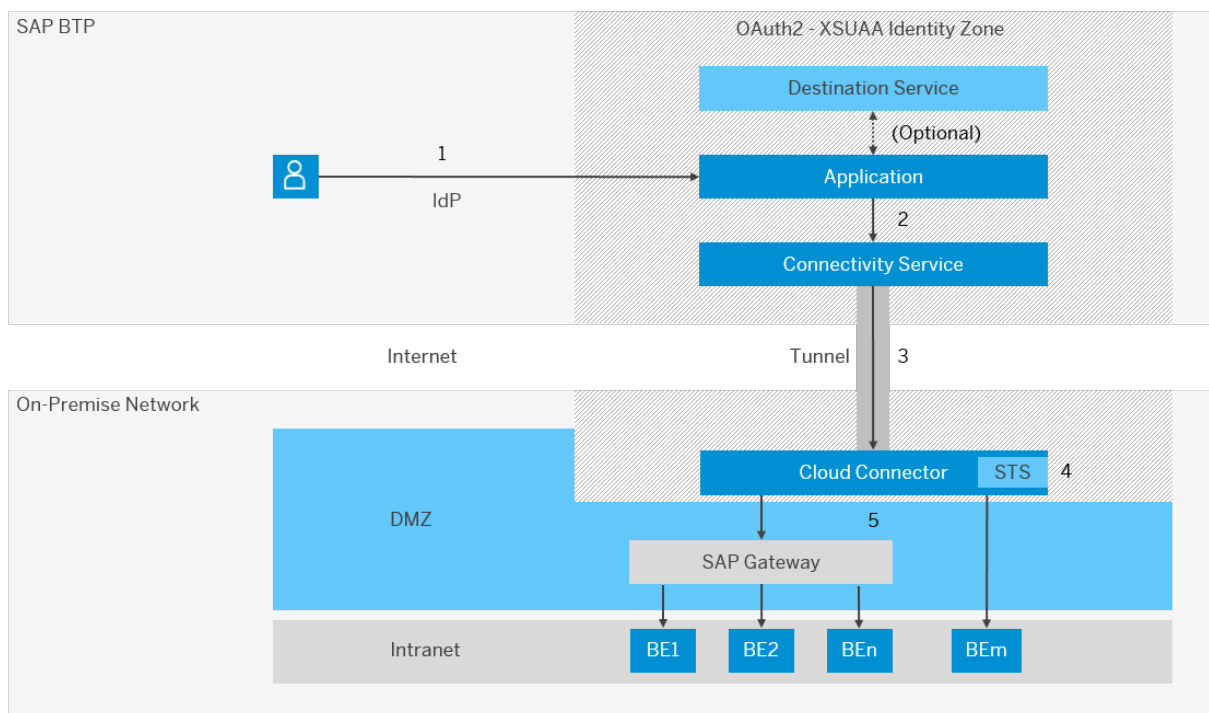
Optionally, you can configure and use a destination configuration by setting the authentication type as `PrincipalPropagation`. For more information, see [Managing Destinations \[page 59\]](#).

Note

This scenario is only applicable if the on-premise system is exposed to the cloud via the Cloud Connector.

You can configure principal propagation for HTTP or RFC communication.



Scenario: Cloud to On-Premise



1. A user logs in to the cloud application. Its identity is established by an identity provider (this can be the default IdP for the subaccount or another trusted IdP).
2. The cloud application then uses a user exchange token (or a designated secondary header) to propagate the user to the Connectivity service. See also [Configure Principal Propagation via User Exchange Token \[page 227\]](#).
 - Optionally, the application may use the Destination service to externalize the connection configuration that points to the target on-premise system. See also [Consuming the Destination Service \[page 243\]](#).
 - If you are using RFC as communication protocol with the *SAP Java Buildpack*, this step is already done by the Java Connector (JCo).

3. The Connectivity service forwards the JWT (that represents the user) to the Cloud Connector.
4. The Cloud Connector receives the JWT, verifies it, extracts the attributes, and uses its STS (security token service) component to issue a new token (for example, an X.509 certificate) with the same or similar attributes to assert the identity to the backend (BE1-BEm). The Cloud Connector and the cloud application share the same trust settings, see [Set Up Trust \[page 420\]](#).
5. The Cloud Connector sends the new token (for example, an X.509 certificate) to the backend system.

Configuration: Cloud to On-Premise

Task Type	Task
 Operator	Configuring Principal Propagation [page 420] (Cloud Connector)
 Developer	Use cases: <ul style="list-style-type: none"> • HTTP communication: Configure Principal Propagation via User Exchange Token [page 227] (Connectivity service) • RFC communication: Configure Principal Propagation for RFC [page 334]

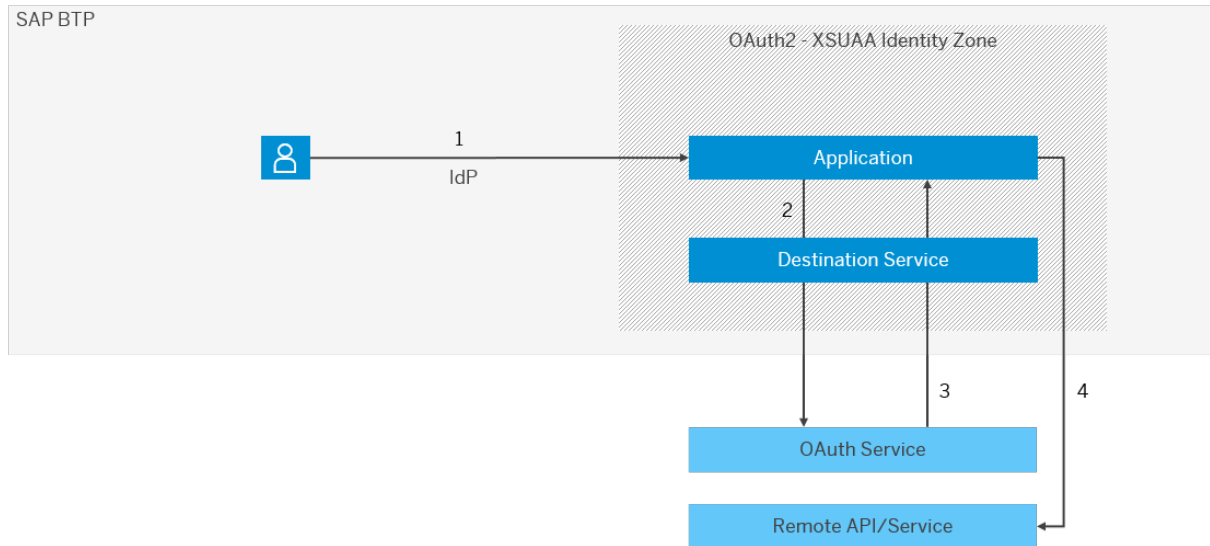
1.1.3.4.2 Scenario: Cloud to Cloud

Forward the identity of cloud users from the Cloud Foundry environment to remote systems on the Internet, enabling single sign-on (SSO).

Concept

The Destination service provides a secure way of forwarding the identity of a cloud user to another remote system or service using a destination configuration with authentication type `OAuth2SAMLBearerAssertion`. This enables the cloud application to consume OAuth-protected APIs exposed by the target remote system.



Scenario: Cloud to Cloud



1. A user logs in to the cloud application. Its identity is established by an identity provider (this can be the default IdP for the subaccount or another trusted IdP).
2. When the application retrieves an `OAuthSAMLBearer` destination, the user is made available to the Destination Service by means of a user exchange JWT. The service then wraps the user identity in a SAML assertion, signs it with the subaccount's private key and sends it to the specified OAuth token service.
3. The OAuth token service accepts the SAML assertion and returns an OAuth access token. In turn, the Destination service returns both the destination and the access token to the requesting application.
4. The application uses the destination properties and the access token to consume the remote API.

You can set up user propagation for connections to applications in different cloud systems or environments.

Configuration: Cloud to Cloud

Task Type	Task
 Operator	Set up Trust Between Systems [page 172]
 Operator and/or Developer	User Propagation via SAML 2.0 Bearer Assertion Flow [page 251] (Destination service)

Use Cases: Cloud to Cloud

- [User Propagation from the Cloud Foundry Environment to SAP S/4HANA Cloud \[page 174\]](#)
- [User Propagation from the Cloud Foundry Environment to SAP SuccessFactors \[page 182\]](#)
- [User Propagation between Cloud Foundry Applications \[page 188\]](#)
- [User Propagation from the Cloud Foundry Environment to the Neo Environment \[page 196\]](#)

1.1.3.4.2.1 Set up Trust Between Systems

Download and configure X.509 certificates as a prerequisite for user propagation from the Cloud Foundry environment.

Setting up a trust scenario for user propagation requires the exchange of public keys and certificates between the affected systems, as well as the respective trust configuration within these systems. This enables you to use an HTTP destination with authentication type `OAuth2SAMLBearerAssertion` for the communication.

A trust scenario can include user propagation from the Cloud Foundry environment to another SAP BTP environment, to another Cloud Foundry subaccount, or to a remote system outside SAP BTP, like S/4HANA Cloud, C4C, Success Factors, and others.

Set Up a Certificate

Download and save locally the identifying X509 certificate of the subaccount in the Cloud Foundry environment.

1. In the cloud cockpit, log on with `Administrator` permission.
2. Navigate to your subaccount in the Cloud Foundry environment.
3. From the left-side menu, choose **Connectivity** > **Destinations**.
4. Choose the **Download Trust** button and save locally the X.509 certificate that identifies this subaccount.

The screenshot shows the SAP BTP Cockpit interface. The left sidebar contains a navigation menu with 'Destinations' selected. The main content area shows 'Subaccount: trial - Destinations' with a search bar and a table of destinations. The 'Download Trust' button is highlighted in red. Below the table, the details for the 'test' destination are visible.

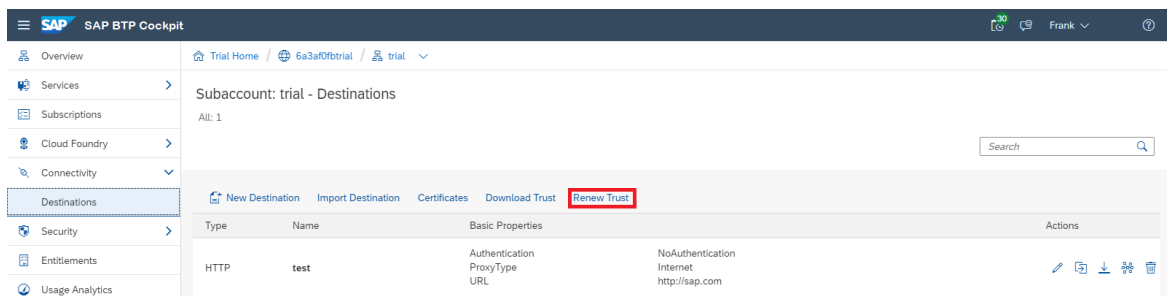
Type	Name	Basic Properties	Actions
HTTP	test	Authentication ProxyType URL	NoAuthentication Internet http://sap.com

5. Configure the downloaded X.509 certificate in the target system to which you want to propagate the user.

Renew a Certificate

If the X.509 certificate validity is about to expire, you can renew the certificate and extend its validity by another 2 years.

1. In the cloud cockpit, log on with Administrator permission.
2. Navigate to your subaccount in the Cloud Foundry environment.
3. From the left-side menu, choose ► **Connectivity** ► **Destinations** ►.
4. Choose the **Renew Trust** button to trigger a renewal of the existing X509 certificate.



5. Choose the **Download Trust** button and save locally the X.509 certificate that identifies this subaccount.
6. Configure the renewed X.509 certificate in the target system to which you want to propagate the user.

Rotate Certificates

You can rotate the identifying X.509 certificate of the subaccount. Rotation is done by creating a passive X.509 certificate for the subaccount, configuring it in the target system to which you want to propagate the user, and rotating it with the active one. After rotation is performed, the active X.509 certificate becomes passive and the passive one active.

Note

The passive X.509 certificate and the certificate rotation can be managed only via the Destination service REST API. For more information, see [Destination Service REST API \[page 80\]](#).

Procedure

1. Generate or renew the passive X.509 certificate: `POST /saml2Metadata/certificate/passive`.
2. Download and save locally the passive X.509 certificate: `GET /saml2Metadata/certificate/passive`.
3. Configure the downloaded X.509 passive certificate in the target system you want to propagate the user to.
4. Rotate the active certificate, making the active one passive and the passive one active: `POST /saml2Metadata/rotateCertificate`.
5. Check that user propagation is working correctly. If it is not, you can rotate the certificates again until fixing the issue.
6. (Optional): Delete the passive X.509 certificate, which used to be active before rotation: `DELETE /saml2Metadata/certificate/passive`.

7. (Optional): Delete the passive X.509 certificate, which used to be active before rotation, from the target system.

Related Information

[Principal Propagation from the Cloud Foundry to the Neo Environment](#)

[User Propagation from the Cloud Foundry Environment to SAP S/4HANA Cloud \[page 174\]](#)

[User Propagation from the Cloud Foundry Environment to SAP SuccessFactors \[page 182\]](#)

[SAP BTP Destination Service Expiring Certificate Notification](#)

1.1.3.4.2.2 User Propagation from the Cloud Foundry Environment to SAP S/4HANA Cloud

Configure user propagation (single sign-on), using OAuth communication from the SAP BTP Cloud Foundry environment to S/4HANA Cloud. As OAuth mechanism, you use the *OAuth 2.0 SAML Bearer Assertion Flow*.

Steps

[Scenario \[page 174\]](#)

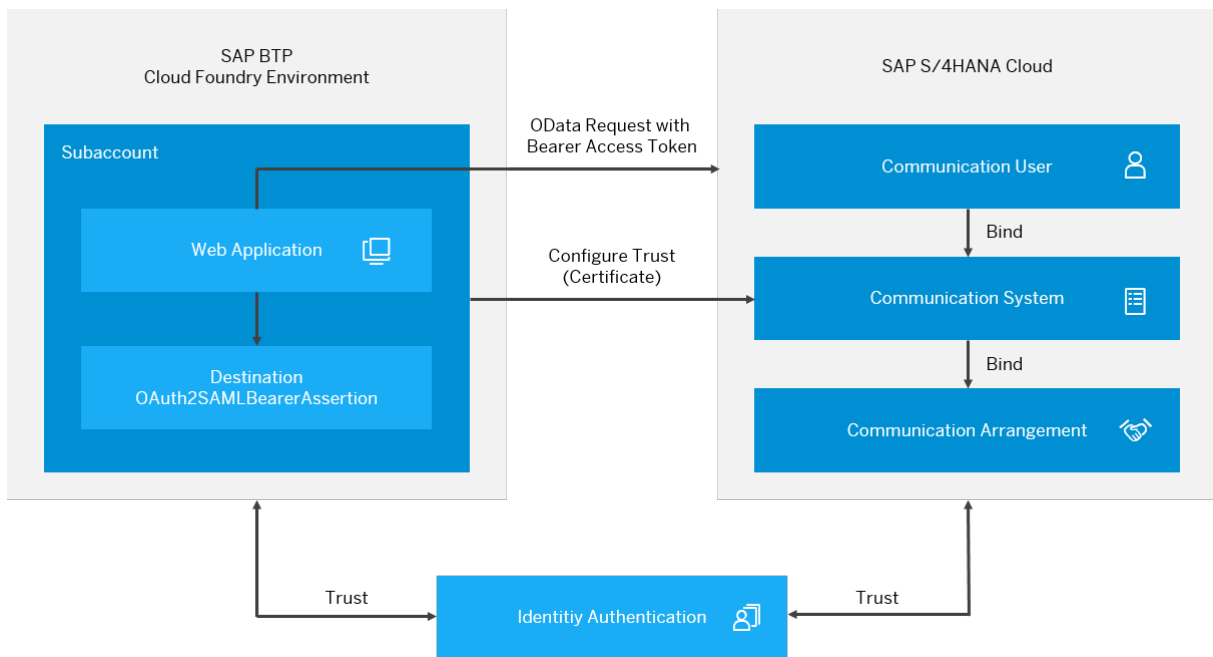
[Prerequisites \[page 175\]](#)

[Configuration Tasks \[page 176\]](#)

Scenario

As a customer, you own an SAP BTP global account and have created at least one subaccount therein. Within the subaccount, you have deployed a Web application. Authentication against the Web application is based on a trusted identity provider (IdP) that you need to configure for the subaccount.

On the S/4HANA Cloud side, you own an S/4HANA ABAP tenant. Authentication against the S/4HANA ABAP tenant is based on the trusted IdP which is always your Identity Authentication Service (IAS) tenant. Typically, you will configure this S/4HANA Cloud Identity tenant to forward authentication requests to your corporate IdP.



Prerequisites

- You have an S/4HANA Cloud tenant and a user with the following business catalogs assigned:

Business Role ID	Area
SAP_BCR_CORE_COM	Communication Management
SAP_BCR_CORE_IAM	Identity and Access Management
SAP_BCR_CORE_EXT	Extensibility

- You have administrator permission for the configured S/4HANA Cloud IAS tenant.
- You have a subaccount and PaaS tenant in the SAP BTP Cloud Foundry environment.

Next Step

- [Configuration Tasks \[page 176\]](#)

1.1.3.4.2.1 Configuration Tasks

Perform these steps to set up user propagation between S/4HANA Cloud and the SAP BTP Cloud Foundry environment.

Tasks

1. [Configure Single Sign-On between S/4HANA Cloud and the Cloud Foundry Organization on SAP BTP \[page 176\]](#)
2. [Configure OAuth Communication \[page 176\]](#)
3. [Configure Communication Settings in S/4HANA Cloud \[page 177\]](#)
4. [Configure Communication Settings in SAP BTP \[page 180\]](#)
5. [Consume the Destination and Execute the Scenario \[page 182\]](#)

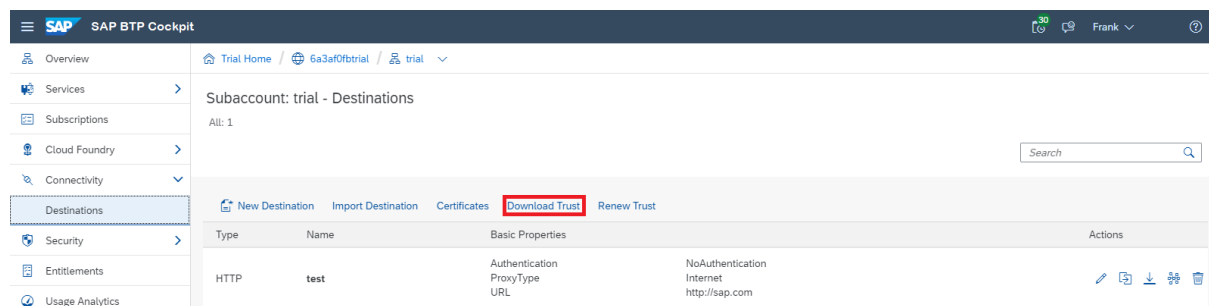
Configure Single Sign-On between S/4HANA Cloud and the Cloud Foundry Organization on SAP BTP

To configure SSO with S/4HANA you must configure trust between the S/4HANA IAS tenant and the Cloud Foundry organization, see [Manually Establish Trust and Federation Between SAP Authorization and Trust Management Service and Identity Authentication](#).

Configure OAuth Communication

Download the certificate from your Cloud Foundry subaccount on SAP BTP.

1. From the SAP BTP cockpit, choose **Cloud Foundry environment** **your global account**.
2. Choose or create a subaccount, and from your left-side subaccount menu, go to **Connectivity** **Destinations**.
3. Press the **Download Trust** button.



The screenshot shows the SAP BTP Cockpit interface. The left sidebar contains a navigation menu with items like Overview, Services, Subscriptions, Cloud Foundry, Connectivity, Destinations, Security, Entitlements, and Usage Analytics. The main content area is titled 'Subaccount: trial - Destinations' and shows a table of destinations. The 'Download Trust' button is highlighted with a red box. Below the table, the basic properties of the 'test' destination are displayed.

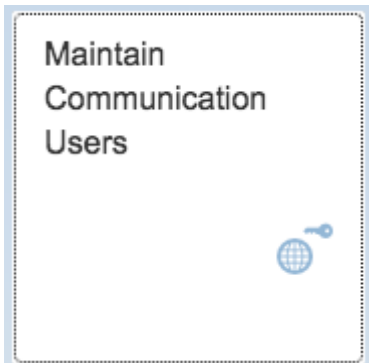
Type	Name	Basic Properties	Actions
HTTP	test	Authentication ProxyType URL	NoAuthentication Internet http://sap.com

Back to [Tasks \[page 176\]](#)

Configure Communication Settings in S/4HANA Cloud

1. Create a Communication User

1. In your S/4HANA Cloud launchpad, choose the application *Maintain Communication Users*.



2. From the *User List* view, create a new user.
3. Set *<User Name>*, *<Password>* and *<Description>*.
4. Copy this password, you will need it in a later step.
5. Press the *Save* on the bottom of the screen.

VIKTOR
User ID: C0000000

User Data

*User Name: *Description:
User ID: C0000000 User Lock Status:

Password

Password:
Password Status: Productive

Certificate

Subject:
Issuer:

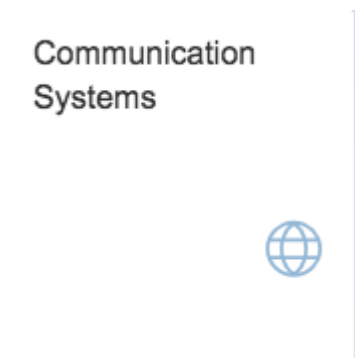
Used by Communication Systems

System ID	System Name	Description	Host Name
HCPEXT_CF	HCPEXT_CF		int.saps.hana.ondemand.com

6. Close the *Communication Users* application.

2. Set up a Communication System for OAuth

1. From the launchpad, choose the application *Communication Systems*.



2. From the list view, select *New*.
3. A popup window appears. Enter the <System ID> and the <System Name>, then choose *Create*.

New Communication System

*System ID:

*System Name:

Create **Cancel**

4. Enter the host name. This is your Cloud Foundry region, for example: `cf.eu10.hana.ondemand.com` for Europe (Frankfurt).

Note

For the complete list of standard regions, see [Regions](#).

General

*Host Name:

Logical System:

HTTPS Port:

5. Enable the OAuth Identity Provider.

OAuth 2.0 Identity Provider

Enabled

Provider Name

Signing Certificate ...

Signing Certificate I...

- Upload the subaccount certificate that you have downloaded before from the SAP BTP cockpit.

OAuth 2.0 Identity Provider

Enabled:

*Provider Name:

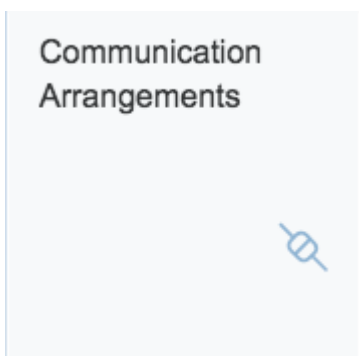
Signing Certificate Subject: OU=CP Destination
Configuration,O=SAP,CN=cfapps.sap.hana.ondemand.com/
a352a17b- <...>

Signing Certificate Issuer: OU=CP Destination
Configuration,O=SAP,CN=cfapps.sap.hana.ondemand.com/
a352a17b- <...>

- From [<Signing Certificate Subject>](#), copy the CN value (for example: `cfapps.sap.hana.ondemand.com/a352a17b-<...>`) and paste it in the field [<Provider Name>](#).
- Add the [Communication User](#) you have created in the previous step.

User for Inbound Communication		+
Authentication Method	User Name	
User ID and Password	VIKTOR	⊗

- Save your settings and go back to the launchpad.
- Create a Communication Arrangement**
 - Start the [Communication Arrangements](#) application.



- From the list view, select [New](#).

- In the popup, choose a scenario. For our example, we use SAP_COM_0013. Set the arrangement name, for example **SAP_COM_0013_MY_TEST**.
- In the *Common Data* section of the configuration screen, select the *<Communication System>* that you have created in the step before. The communication user is added automatically in the *Inbound Communication* section, and the *<Authentication Method>* is set to OAuth 2.0.

SAP_COM_0013_CV_CF

Scenario ID: SAP_COM_0013 Scenario Description: SAP Web IDE Integration Draft Last Changed By: Draft Last Changed On: 12/01/2017, 12:42:01

Editing Status: Draft

Common Data

Arrangement Name: Own System:

*Communication System:

Inbound Communication

*User Name: Authentication Method:

Inbound Services

Service	Application Protocol	Service URL / Service Interface	WSDL	Additional Properties
Catalog Service Version 2	OData V2	https://my300117-api.s4hana.ondemand.com/sap/opu/odata/IWFND/CATALOGSERVICE;v=2		
Gateway service for ADT	OData V2	https://my300117-api.s4hana.ondemand.com/sap/opu/odata/sap/ADT_SRV		
UI2 App Index Services	OData V2			

- In the *Outbound Services* section, go to *Launch SAP Web IDE* and uncheck the *Active* checkbox of the field *<Service Status>*.

Outbound Services

▼ Launch SAP Web IDE

Service Status: Active

Path:

Application Protocol: Service URL:

Port:

- Save your settings and go back to the launchpad.

Back to [Tasks \[page 176\]](#)

Configure Communication Settings in SAP BTP

- From the SAP BTP cockpit, choose ► *Cloud Foundry environment* ► *your global account* ►.
- Choose your subaccount, and from the left-side subaccount menu, go to ► *Connectivity* ► *Destinations* ►.
- Press the *New Destination* button.
- Enter the following parameters for your destination:

Parameter	Value
Name	Enter a meaningful name.
Type	HTTP
Description	(Optional) Enter a meaningful description.
URL	The OData URL, for example <code>https://my300117-api.s4hana.ondemand.com/sap/opu/odata/IWFND/CATALOGSERVICE;v=2?\$format=json</code>
Proxy Type	Internet
Authentication	OAuth2SAMLBearerAssertion
Audience	<p>The URL of your SAP S/4HANA Cloud account.</p> <p>To get it, log on to your SAP S/4HANA Cloud account. Select the profile picture. Then choose <i>Settings</i> and copy the value from the <code><Server></code> field. Add <code>https://</code> to the beginning of this string, for example, <code>https://my300117.s4hana.ondemand.com</code>.</p> <div style="border: 1px solid #ccc; background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note</p> <p>This URL does not contain <code>my300117-api</code>, but only <code>my300117</code>.</p> </div>
Client Key	The name of the communication user you have in the SAP S/4HANA ABAP tenant, e.g VIKTOR.
Token Service URL	<p>For this field, you need the part of the URL before <code>/sap/...</code> that you copied before from <i>Communications Arrangements</i> service URL/service interface:</p> <p><code>https://my300117-api.s4hana.ondemand.com/sap/bc/sec/oauth2/token</code></p>
Token Service User	The same user as for the Client Key parameter.
Token Service Password	The password for the communication user.
System User	This parameter is not used, leave the field empty.
authnContextClassRef	urn:oasis:names:tc:SAML:2.0:ac:classes:x509

Destination Configuration

*Name:

Type:

Description:

*URL:

Proxy Type:

Authentication:

*Audience:

*Client Key:

*Token Service URL:

Token Service User:

Token Service Password:

System User:

Additional Properties

Use default JDK truststore

Back to [Tasks \[page 176\]](#)

Consume the Destination and Execute the Scenario

To perform the scenario and execute the request from the source application towards the target application, proceed as follows:

1. Decide on where the user identity will be located when calling the Destination service. For details, see [User Propagation via SAML 2.0 Bearer Assertion Flow \[page 251\]](#). This will determine how exactly you will perform step 2.
2. Execute a "find destination" request from the source application to the Destination service. For details, see [Consuming the Destination Service \[page 243\]](#) and the [REST API documentation](#).
3. From the Destination service response, extract the access token and URL, and construct your request to the target application. See ["Find Destination" Response Structure \[page 259\]](#) for details on the structure of the response from the Destination service.

Back to [Tasks \[page 176\]](#)

1.1.3.4.2.3 User Propagation from the Cloud Foundry Environment to SAP SuccessFactors

Configure user propagation from the SAP BTP Cloud Foundry environment to SAP SuccessFactors.

Steps

[Scenario \[page 183\]](#)

[Prerequisites \[page 183\]](#)

[Concept Overview \[page 183\]](#)

[Create an OAuth Client in SAP SuccessFactors \[page 184\]](#)

[Create and Consume a Destination for the Cloud Foundry Application \[page 186\]](#)

Scenario

- From an application in the SAP BTP Cloud Foundry environment, you want to consume OData APIs exposed by SuccessFactors modules.
- To enable single sign-on, you want to propagate the identity of the application's logged-in user to SuccessFactors.

Prerequisites

- In your Cloud Foundry space, you have a deployed application.
- You have an instance of the Destination Service that is bound to the application.
- An instance of the xsuaa service with `application` plan is bound to the application.

Concept Overview

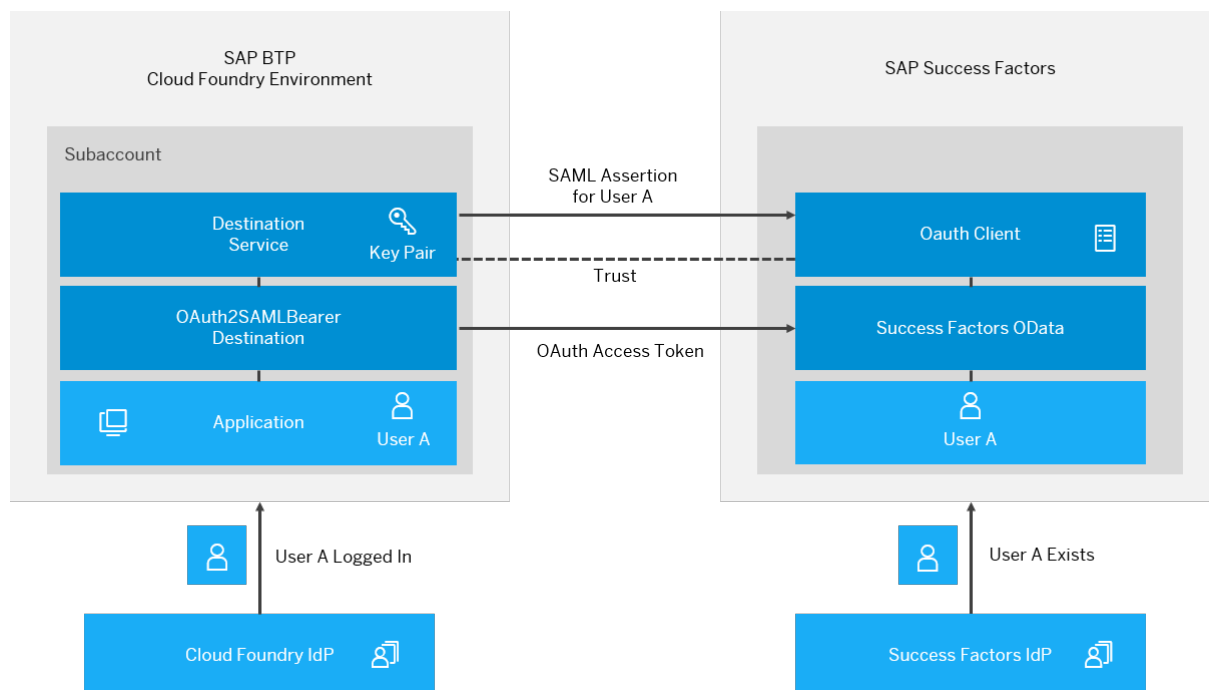
A user logs in to the Cloud Foundry application. Its identity is established by an **Identity Provider** (IdP). This could be the default IdP for the Cloud Foundry subaccount or a trusted IdP, for example the SuccessFactors IdP.

When the application retrieves an `OAuth2SAMLBearer` destination, the user is made available to the Cloud Foundry Destination service by means of a **user exchange token**, represented by a JSON Web Token (JWT). The service then wraps the user identity in a SAML assertion, signs it with the Cloud Foundry subaccount private key and sends it to the token endpoint of the SuccessFactors OAuth server.

To accept the SAML assertion and return an **access token**, a **trust** relationship must be set up between SuccessFactors and the Cloud Foundry subaccount public key. You can achieve this by providing the Cloud Foundry subaccount **X.509 certificate** when creating the OAuth client in SuccessFactors.

Users that are propagated from the Cloud Foundry application, are verified by the SuccessFactors OAuth server before granting them access tokens. This means, users that do not exist in the SuccessFactors user store will be rejected.

For valid users, the OAuth server accepts the SAML assertion and returns an OAuth access token. In turn, the Destination service returns both the destination and the access token to the requesting application. The application then uses the destination properties and the access token to consume SuccessFactors APIs.



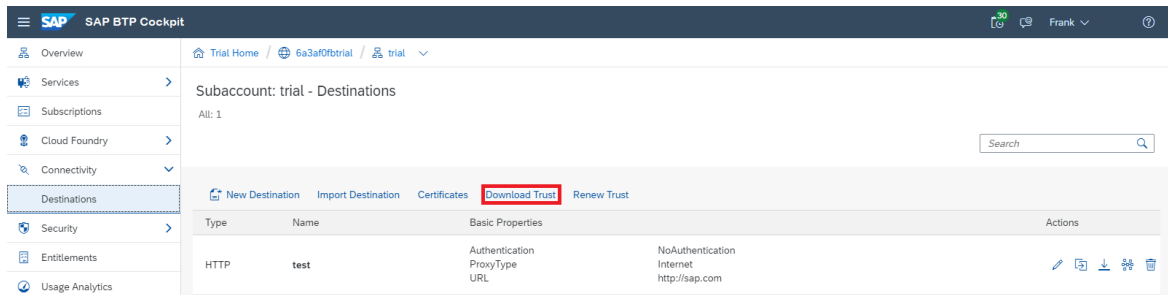
Next Steps

- [Create an OAuth Client in SAP SuccessFactors \[page 184\]](#)
- [Create and Consume a Destination for the Cloud Foundry Application \[page 186\]](#)

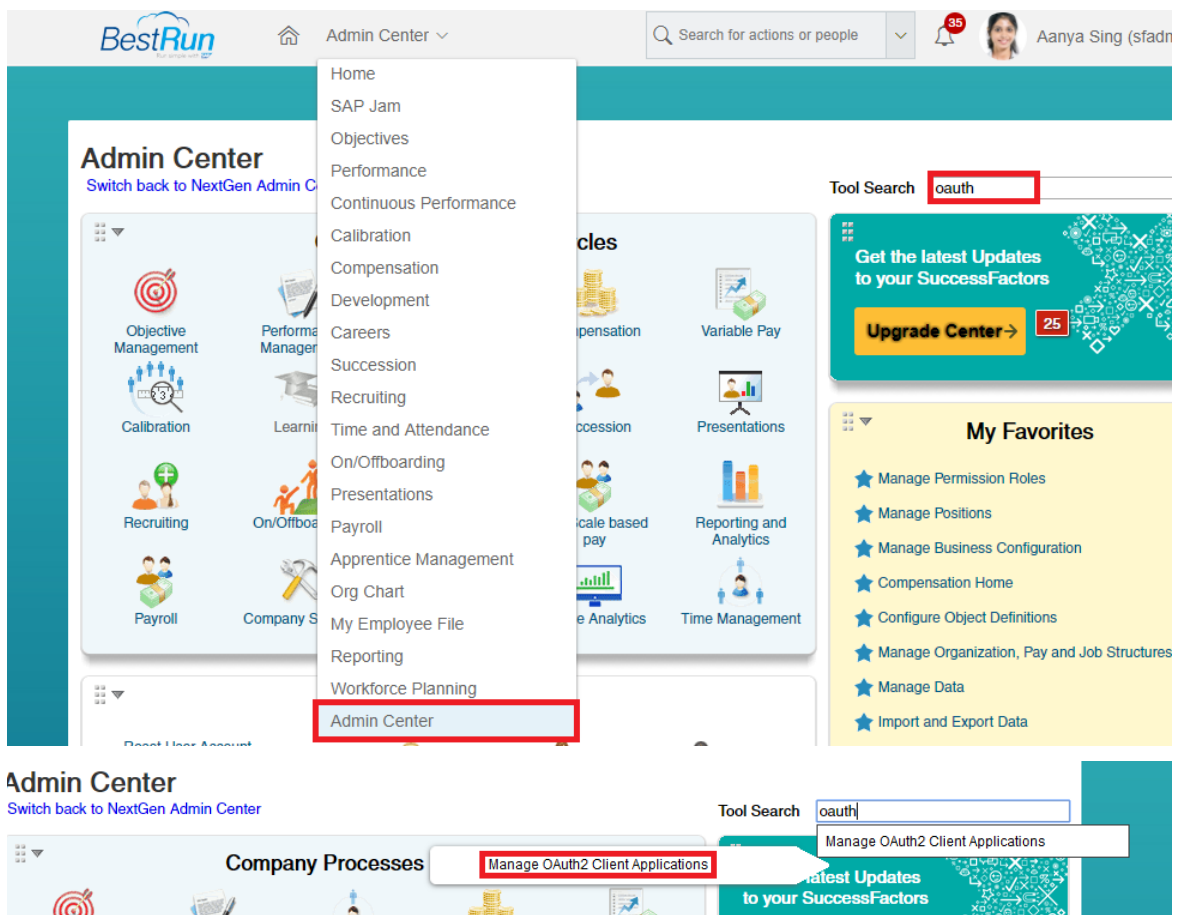
1.1.3.4.2.3.1 Create an OAuth Client in SAP SuccessFactors

Create an OAuth client in SuccessFactors for user propagation from the SAP BTP Cloud Foundry environment.

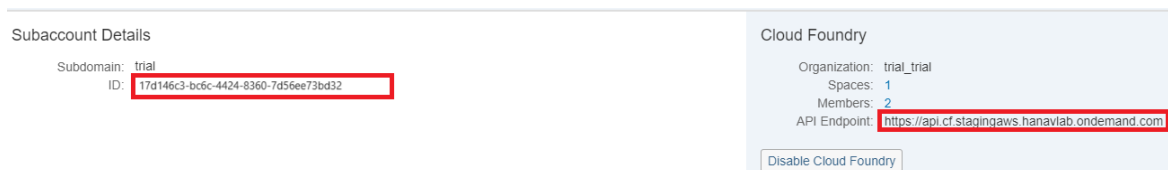
1. Download the X.509 certificate from your Cloud Foundry subaccount:
In the cloud cockpit, navigate to your Cloud Foundry subaccount and from the left-side subaccount menu, choose **Connectivity > Destinations**. Choose **Download Trust** to get the certificate for this subaccount.



2. Create a SuccessFactors OAuth Client:
In SuccessFactors, go to the *Admin Center* and search for **OAuth**. Choose **Manage OAuth2 Client Applications**.



3. Press the *Register Client Application* button on the right. In the *<Application Name>* field, provide some arbitrary descriptive name for the client. For *<Application URL>*, enter the Cloud Foundry API endpoint of the client SAP BTP subaccount, followed by the subaccount GUID, for example `https://api.cf.stagingaws.hanavlab.ondemand.com/17d146c3-bc6c-4424-8360-7d56ee73bd32`. This information is available in the cloud cockpit under subaccount details:



4. In the field *<x.509 Certificate>*, paste the certificate that you downloaded in step 1.

- **<URL>**: URL of the SuccessFactors OData API you want to consume.
 - **<Authentication>**: **OAuth2SAMLBearerAssertion**
 - **<Audience>**: **www.successfactors.com**
 - **<Client Key>**: API Key of the OAuth client you created in SuccessFactors.
 - **<Token Service URL>**: API endpoint URL for the SuccessFactors instance, followed by /
oauth/token and the URL parameter company_id with the company ID, for example https://
apisalesdemo2.successfactors.eu/oauth/token?company_id=SFPART019820.
2. Enter three additional properties:
- **apiKey**: the API Key of the OAuth client you created in SuccessFactors.
 - **authnContextClassRef**: **urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession**
 - **nameIdFormat**:
 - **urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified**, if the **user ID** will be propagated to a SuccessFactors application, or
 - **urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress**, if the **user e-mail** will be propagated to SuccessFactors.

Destination Configuration

<p>*Name: <input type="text" value="sf_serious_destination"/></p> <p>Type: <input type="text" value="HTTP"/></p> <p>Description: <input type="text"/></p> <p>*URL: <input type="text" value="https://salesdemo.successfactors.eu/oauth/token?company_id=SFPART019820"/></p> <p>Proxy Type: <input type="text" value="Internet"/></p> <p>Authentication: <input type="text" value="OAuth2SAMLBearerAssertion"/></p> <p>*Audience: <input type="text" value="www.successfactors.com"/></p> <p>*Client Key: <input type="text" value="*****"/></p> <p>*Token Service URL: <input type="text" value="https://salesdemo.successfactors.eu/oauth/token?company_id=SFPART019820"/></p> <p>Token Service User: <input type="text"/></p> <p>Token Service Password: <input type="text"/></p> <p>System User: <input type="text"/></p>	<p>Additional Properties</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;"><input type="text" value="apiKey"/></td> <td style="width: 50%;"><input type="text" value="ZGNmNDNhMwI5Nj..."/></td> </tr> <tr> <td><input type="text" value="authnContextCla..."/></td> <td><input type="text" value="urn:oasis:names:tc:S..."/></td> </tr> <tr> <td><input type="text" value="nameIdFormat"/></td> <td><input type="text" value="urn:oasis:names:tc:S..."/></td> </tr> </table> <p><input checked="" type="checkbox"/> Use default JDK truststore</p>	<input type="text" value="apiKey"/>	<input type="text" value="ZGNmNDNhMwI5Nj..."/>	<input type="text" value="authnContextCla..."/>	<input type="text" value="urn:oasis:names:tc:S..."/>	<input type="text" value="nameIdFormat"/>	<input type="text" value="urn:oasis:names:tc:S..."/>
<input type="text" value="apiKey"/>	<input type="text" value="ZGNmNDNhMwI5Nj..."/>						
<input type="text" value="authnContextCla..."/>	<input type="text" value="urn:oasis:names:tc:S..."/>						
<input type="text" value="nameIdFormat"/>	<input type="text" value="urn:oasis:names:tc:S..."/>						

Consume the Destination and Execute the Scenario

To perform the scenario and execute the request from the source application towards the target application, proceed as follows:

1. Decide on where the user identity will be located when calling the Destination service. For details, see [User Propagation via SAML 2.0 Bearer Assertion Flow \[page 251\]](#). This will determine how exactly you will perform step 2.
2. Execute a "find destination" request from the source application to the Destination service. For details, see [Consuming the Destination Service \[page 243\]](#) and the [REST API documentation](#).
3. From the Destination service response, extract the access token and URL, and construct your request to the target application. See ["Find Destination" Response Structure \[page 259\]](#) for details on the structure of the response from the Destination service.

1.1.3.4.2.4 User Propagation between Cloud Foundry Applications

Propagate the identity of a user between Cloud Foundry applications that are located in different subaccounts or regions.

Steps

[Scenario \[page 188\]](#)

[Prerequisites \[page 188\]](#)

[Concept \[page 189\]](#)

[Procedure \[page 191\]](#)

1. [Assemble IdP Metadata for Subaccount 1 \[page 191\]](#)
2. [Establish Trust between Subaccount 1 and Subaccount 2 \[page 192\]](#)
3. [Create an OAuthSAMLBearerAssertion Destination for Application 1 \[page 193\]](#)
4. [Consume the Destination and Execute the Scenario \[page 195\]](#)

Scenario

- You have deployed an application in a Cloud Foundry environment (**application 1**).
- You want to call another Cloud Foundry application (**application 2**) in a different subaccount, in the same or another region.
- You want to propagate the identity of the user that is logged in to application 1, to application 2.

Back to [Steps \[page 188\]](#)

Prerequisites

- You have two applications (application 1 and application 2) deployed in Cloud Foundry spaces in different subaccounts in the same region or even in different regions.
- You have an instance of the Destination service bound to application 1.
- You have a user JWT (JSON Web Token) in application 1 where the call to application 2 is performed.

Back to [Steps \[page 188\]](#)

Concept

The identity of a user logged in to application 1 is established by an identity provider (IdP) of the respective subaccount (**subaccount 1**).

Note

You can use the default IdP for the Cloud Foundry subaccount or a custom-configured IdP.

When the application retrieves an OAuthSAMLBearer destination, the user is made available to the Cloud Foundry Destination service by means of a *user exchange* JWT. See also [User Propagation via SAML 2.0 Bearer Assertion Flow \[page 251\]](#).

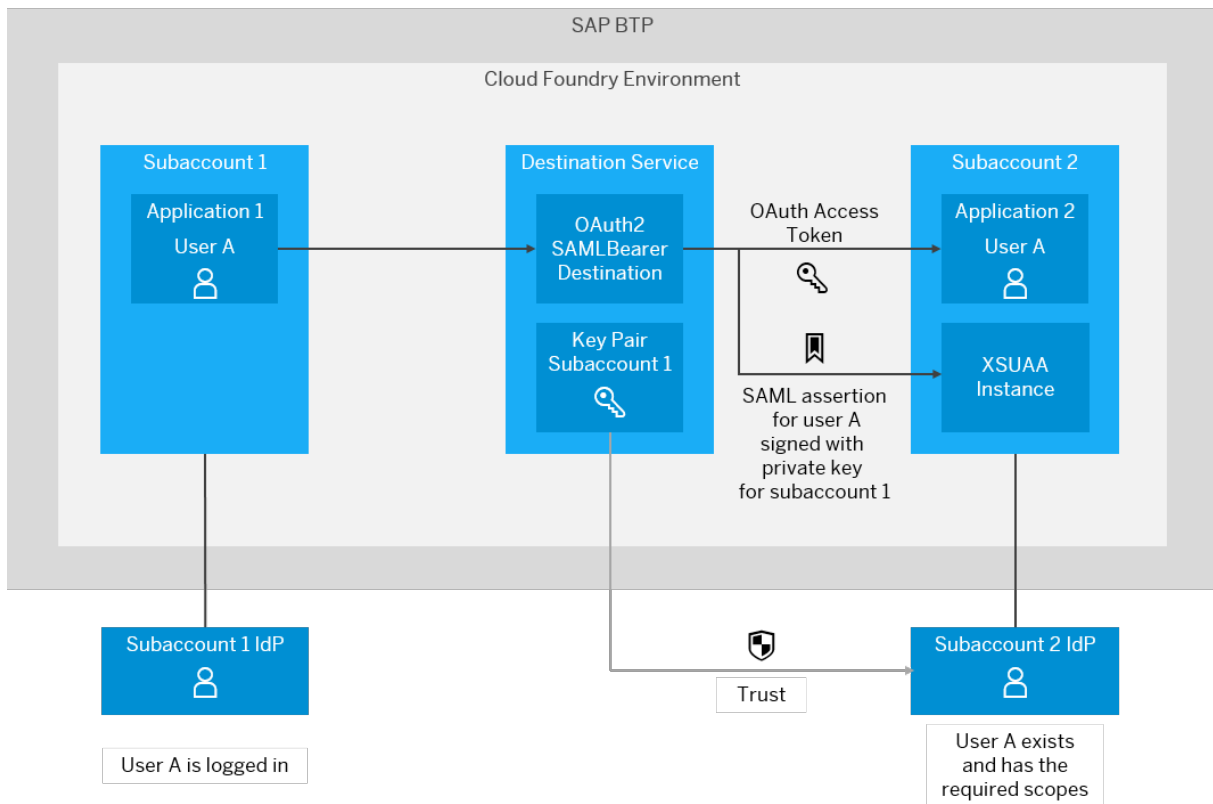
The service then wraps the user identity in a SAML assertion, signs it with subaccount 1's private key (which is part of the special key pair for the subaccount, maintained by the Destination service) and sends it to the authentication endpoint of **subaccount 2**, which hosts application 2.

To make the authentication endpoint accept the SAML assertion and return an access token, you must set up a trust relationship between the two subaccounts, by using subaccount 1's public key. You can achieve this by assembling the SAML IdP metadata, using subaccount 1's public key and setting up a new trust configuration for subaccount 2, which is based on that metadata.

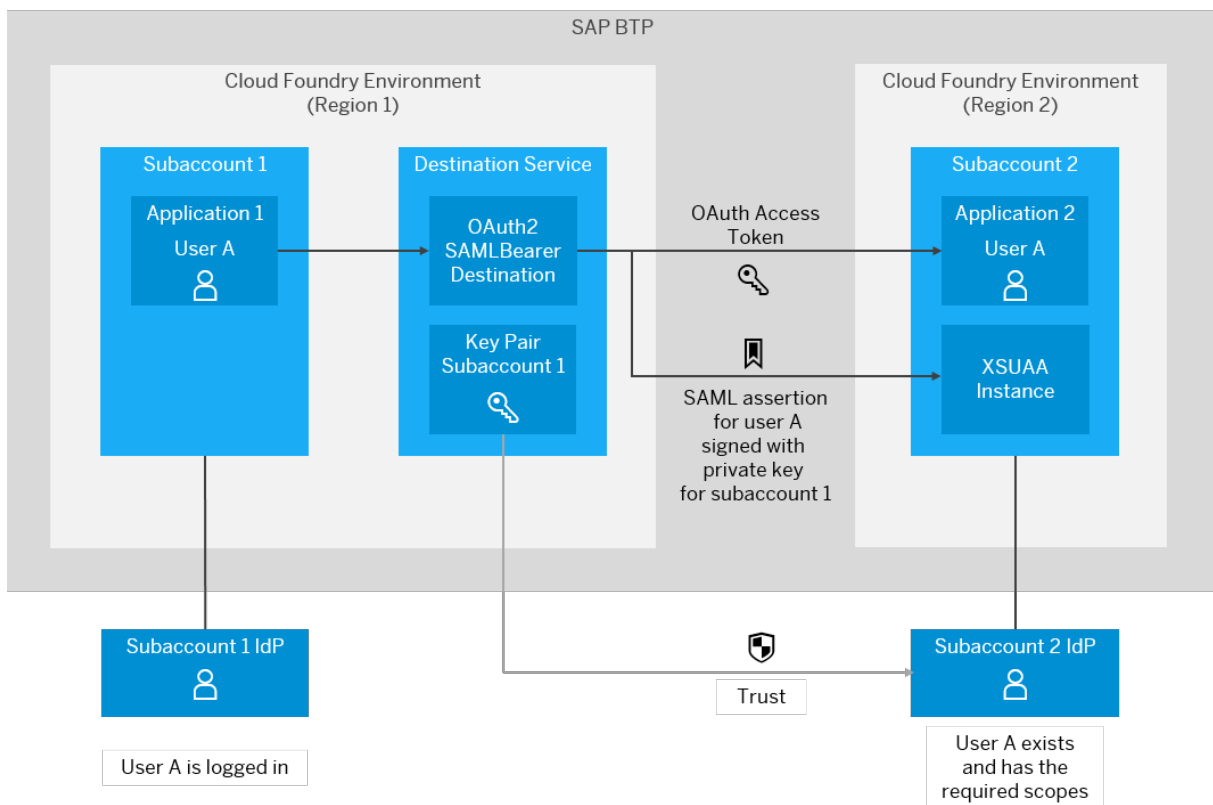
This way, users propagated from application 1 can be verified by subaccount 2's IdP before granting them access tokens with their respective scopes in the context of subaccount 2.

The authentication endpoint accepts the SAML assertion and returns an OAuth access token. In turn, the Destination service returns both the destination configuration and the access token to the requesting application (application 1). Application 1 then uses the destination properties and the access token to call application 2.

Option 1 - Setting up Trust between Subaccounts in the Same Region



Option 2 - Setting up Trust between Subaccounts in Different Regions



Back to [Steps \[page 188\]](#)

Procedure

Assemble IdP Metadata for Subaccount 1

1. Download the X.509 certificate of subaccount 1. For instructions, see [Set up Trust Between Systems \[page 172\]](#). The content of the file is shown as:

```
-----BEGIN CERTIFICATE-----<content>-----END CERTIFICATE-----
```

Below, we refer to the value of <content> as `${S1_CERTIFICATE}`.

2. In the cockpit, navigate to the overview page of subaccount 1. For details, see [Navigate in the Cockpit](#). Here you can see the landscape domain, subaccount ID and subdomain. Below, we refer to the landscape domain as `${S1_LANDSCAPE_DOMAIN}`, to the subaccount ID as `${S1_SUBACCOUNT_ID}` and to the subdomain as `${S1_SUBDOMAIN}`.

The screenshot shows the SAP BTP Cockpit interface for subaccount 1. The left sidebar contains navigation options: Overview, Services, Subscriptions, Cloud Foundry, Connectivity, Destinations, Security, Entitlements, and Usage Analytics. The main content area displays 'Subaccount: 1' with a 'Delete Subaccount' button. Below this, there are fields for 'Subdomain' and 'ID', both highlighted with red boxes. To the right, it shows '1 (24 available) Active Subscriptions'. Underneath, there are tabs for 'Cloud Foundry Environment', 'Kyma Environment', and 'Entitlements'. The 'Cloud Foundry Environment' tab is active, showing 'Org Name: 6a3af0fbtrial', 'Org ID: 3a593519-9d5c-4b16-aaa7-6d3b2ea60d5f', 'Members: 1', and 'API Endpoint: https://api.cf.eu10.hana.ondemand.com', with the endpoint highlighted in red. A 'landscape domain' label is also present. A 'Spaces (2)' table is shown with columns for Name, Applications, and Service Instances. The table contains two rows: 'dev' with 1 application and 3 service instances, and 'trialfm' with 0 applications and 0 service instances. A 'Create Space' button is located at the top right of the table.

3. In your browser, call `https://${S1_SUBDOMAIN}.authentication.${S1_LANDSCAPE_DOMAIN}/saml/metadata` and download the XML file. Within the XML file you can find the following structure:

Sample Code

```
<?xml version="1.0" encoding="UTF-8"?>
...
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
Location="https://${S1_SUBDOMAIN}.authentication.${S1_LANDSCAPE_DOMAIN}/
oauth/token/alias/<alias>" index="1"/>
...
```

Below, we refer to the value of <alias> as `${S1_ALIAS}`.

4. Assemble the new IdP metadata for subaccount 1 by replacing the `${ . . . }` placeholders in the following template with the values determined in the previous steps:

Sample Code

```
<ns3:EntityDescriptor
ID="cfapps.${S1_LANDSCAPE_DOMAIN}/${S1_SUBACCOUNT_ID}"
entityID="cfapps.${S1_LANDSCAPE_DOMAIN}/${S1_SUBACCOUNT_ID}"
```

```

xmlns="http://www.w3.org/2000/09/xmldsig#"
xmlns:ns2="http://www.w3.org/2001/04/xmlenc#"
xmlns:ns4="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:ns3="urn:oasis:names:tc:SAML:2.0:metadata">
<ns3:SPSSODescriptor AuthnRequestsSigned="true"
  protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
  <ns3:KeyDescriptor use="signing">
    <KeyInfo>
      <KeyName>${S1_ALIAS}</KeyName>
      <X509Data>
        <X509Certificate>
          ${S1_CERTIFICATE}
        </X509Certificate>
      </X509Data>
    </KeyInfo>
  </ns3:KeyDescriptor>
</ns3:SPSSODescriptor>
<ns3:IDPSSODescriptor
  WantAuthnRequestsSigned="true"
  protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
  <ns3:KeyDescriptor use="signing">
    <KeyInfo>
      <KeyName>${S1_ALIAS}</KeyName>
      <X509Data>
        <X509Certificate>
          ${S1_CERTIFICATE}
        </X509Certificate>
      </X509Data>
    </KeyInfo>
  </ns3:KeyDescriptor>
</ns3:IDPSSODescriptor>
</ns3:EntityDescriptor>

```

Back to [Steps \[page 188\]](#)

Establish Trust between Subaccount 1 and Subaccount 2

1. In the cockpit, navigate to the overview page for subaccount 2.
2. From the left panel, select **Security** > **Trust Configuration**. Choose *New Trust Configuration*. For details, see [Establish Trust and Federation with UAA Using Any SAML Identity Provider](#).
3. Paste the assembled IdP metadata for subaccount 1 in the `<Metadata>` text box and uncheck *Available for User Logon*.

New Trust Configuration

Metadata: *

```
9exej/8vILgRH8HUffinxbGxh1mPy100kUHG9bDvgykvNxxwJ6hQL5cEh1WipOYtp
tNUipfreWwPkpwAVY97SjuqH/u4Bgvq30+sVPDszS+B1v5XD1S6RMtKB332K0Z7I
a9XHZK9jpwCIMcSSHTSdENN8nvQX59CVuwU+S38V4q2JJ7HUIPNmtTS51QYzPF
kV
8p0823u5+NgLJJA8fbWrhlwgDS++oZAqITffw==
  </X509Certificate>
  </X509Data>
  </KeyInfo>
  </ns3:KeyDescriptor>
  </ns3:IDPSSODescriptor>
  </ns3:EntityDescriptor>
```

Name: *

Description:

Origin Key: *

Status: ▾

Available for User Logon:

Link Text for User Logon:

Create Shadow Users During Logon:

[Show Details](#)

[Parse](#) [Save](#) [Cancel](#)

4. Choose [Parse](#).
5. Enter a [Name](#) for the trust configuration and choose [Save](#).

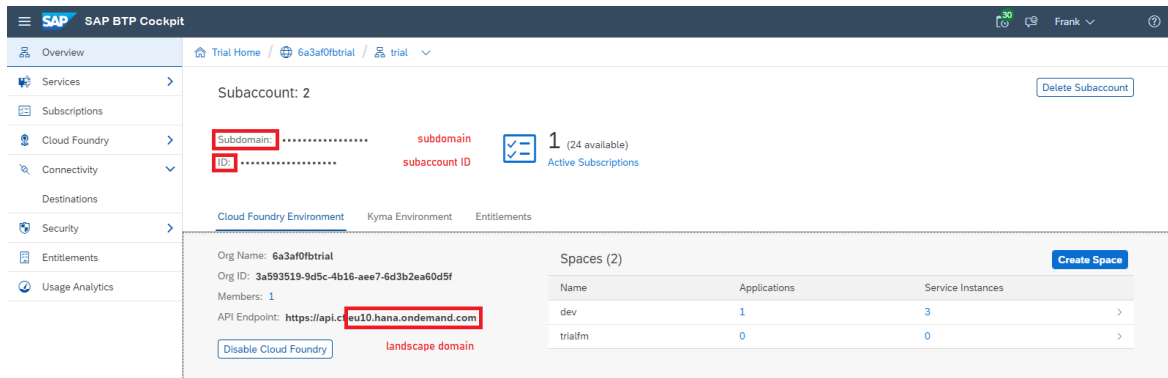
Note

Additionally, you must add users to this new trust configuration and assign appropriate scopes to them.

Back to [Steps \[page 188\]](#)

Create an OAuthSAMLBearerAssertion Destination for Application 1

1. In the cockpit, navigate to the overview page for subaccount 2.
2. Here you can see the landscape domain, subaccount ID and subdomain of subaccount 2. Below, we refer to the landscape domain as `${S2_LANDSCAPE_DOMAIN}`, to the subaccount ID as `${S2_SUBACCOUNT_ID}` and to the subdomain as `${S2_SUBDOMAIN}`.



- In your browser, call `https://${S2_SUBDOMAIN}.authentication.${S2_LANDSCAPE_DOMAIN}/saml/metadata` and download the XML file. Within the XML file, you can find the following structure. It contains the `<audience>` and the `<alias>` variables:

```

{ } Sample Code

<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor entityID="<audience>" ...>
  ...
  <md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI" Location="https://
${S2_SUBDOMAIN}.authentication.${S2_LANDSCAPE_DOMAIN}/oauth/token/alias/
<alias>" index="1"/>
  ...

```

Below, we refer to the value of `<alias>` as `${S2_ALIAS}` and `<audience>` as `${S2_AUDIENCE}`.

- In cockpit, navigate to subaccount 1.
- From the left panel, select **Connectivity** **Destinations**.
- Choose **New Destination** and configure the values as described below. Replace the `${...}` placeholders with the values you determined in the previous steps and sections.

Property	Value
Name	Choose any name for your destination. You will use this name to request the destination from the Destination service.
Type	HTTP
URL	The URL of application 2, identifying the resource you want to consume.
Proxy Type	Internet
Authentication	OAuth2SAMLBearerAssertion
Audience	<code>\${S2_AUDIENCE}</code>
Client Key	The <i>clientid</i> of the XSUAA instance in subaccount 2. Can be acquired via a binding or service key.

Property	Value
Token Service URL	<code>https://\${S2_SUBDOMAIN}.authentication.\${S2_LANDSCAPE_DOMAIN}/oauth/token/alias/\${S2_ALIAS}</code>
Token Service URL Type	Dedicated
Token Service User	The <i>clientid</i> of the XSUAA instance in subaccount 2. Can be acquired via a binding or service key.
Token Service Password	The <i>clientsecret</i> of the XSUAA instance in subaccount 2. Can be acquired via a binding or service key.
authnContextClassRef	urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession

Additional Properties

Property	Value
nameIdFormat	urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

Example

Destination Configuration

*Name:

Type:

Description:

*URL:

Proxy Type:

Authentication:

Key Store Location:

[Upload and Delete Certificates](#)

Key Store Password:

*Audience:

*Client Key:

*Token Service URL:

*Token Service URL Type:

Token Service User:

Token Service Password:

System User:

Additional Properties

Use default JDK truststore

7. Choose [Save](#).

Back to [Steps \[page 188\]](#)

Consume the Destination and Execute the Scenario

To perform the scenario and execute the request from application 1, targeting application 2, proceed as follows:

1. Decide on where the user identity will be located when calling the Destination service. For details, see [User Propagation via SAML 2.0 Bearer Assertion Flow \[page 251\]](#). This will determine how exactly you will perform step 2.
2. Execute a "find destination" request from application 1 to the Destination service. For details, see [Consuming the Destination Service \[page 243\]](#) and the [REST API documentation](#).
3. From the Destination service response, extract the access token and URL, and construct your request to application 2. See ["Find Destination" Response Structure \[page 259\]](#) for details on the structure of the response from the Destination service.

Back to [Steps \[page 188\]](#)

1.1.3.4.2.5 User Propagation from the Cloud Foundry Environment to the Neo Environment

Propagate the identity of a user from a Cloud Foundry application to a Neo application.

Steps

[Scenario \[page 196\]](#)

[Prerequisites \[page 197\]](#)

[Concept \[page 197\]](#)

[Procedure \[page 198\]](#)

1. [Configure a Local Service Provider for the Neo Subaccount \[page 198\]](#)
2. [Establish Trust between Cloud Foundry and Neo Subaccounts \[page 199\]](#)
3. [Create an OAuth Client for the Neo Application \[page 200\]](#)
4. [Create an OAuth2SAMLBearerAssertion Destination for the Cloud Foundry Application \[page 201\]](#)
5. [Consume the Destination and Execute the Scenario \[page 203\]](#)

Scenario

- You have deployed an application in the Cloud Foundry environment.
- You want to consume OAuth protected APIs exposed by an application deployed in the Neo environment.
- You want to propagate the identity of the user logged in to the Cloud Foundry application, to the Neo application.

Back to [Steps \[page 196\]](#)

Prerequisites

- You have deployed an application in the Cloud Foundry environment.
- You have bound an instance of the Destination Service to the application.
- You have bound an instance of the XSUAA service with the `application` plan to the application.
- You have deployed an application in the Neo environment.

Back to [Steps \[page 196\]](#)

Concept

The identity of a user logged in to the Cloud Foundry application is established by an identity provider (IdP).

Note

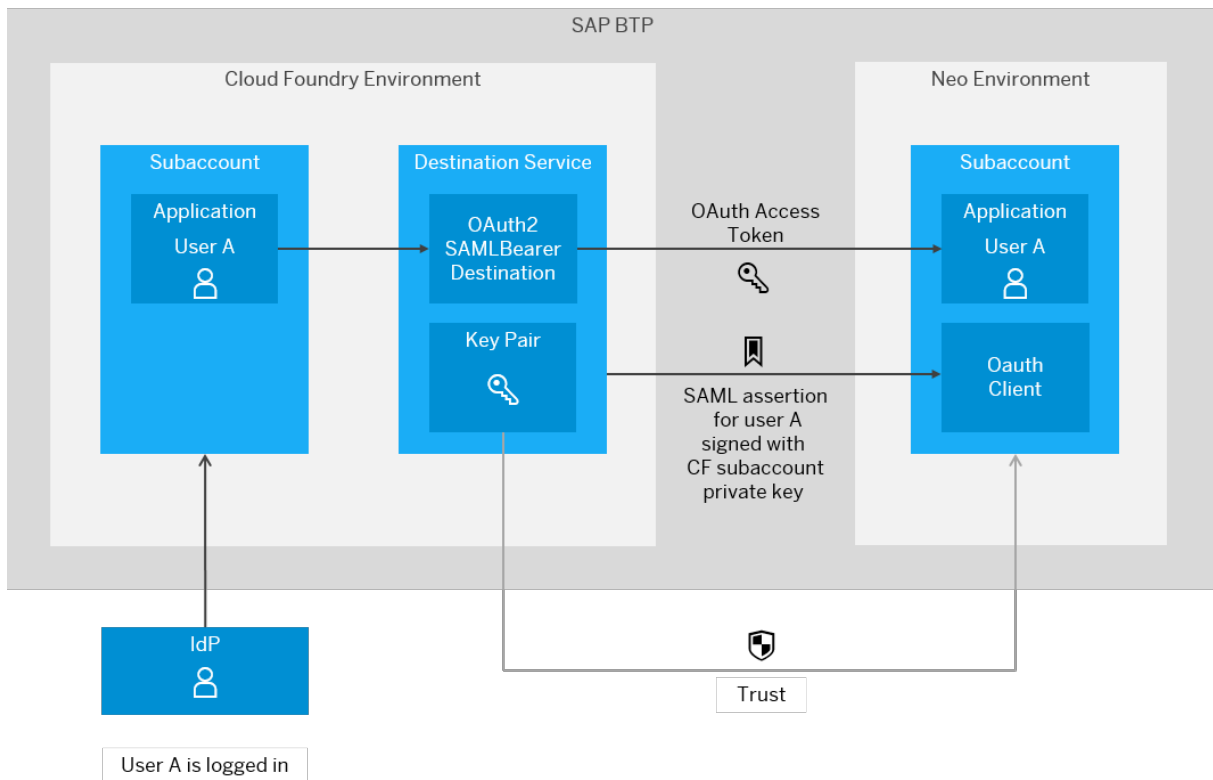
You can use the default IdP of the Cloud Foundry subaccount or any trusted IdP, for example, the Neo subaccount IdP.

When the application retrieves an OAuthSAMLBearer destination, the user is made available to Cloud Foundry Destination service by means of a *user exchange* JWT (JSON Web Token).

The service then wraps the user identity in a SAML assertion, signs it with the Cloud Foundry subaccount private key and sends it to the token endpoint of the OAuth service for the Neo application.

To make the Neo application accept the SAML assertion, you must set up a trust relationship between the Neo subaccount and the Cloud Foundry subaccount public key. You can achieve this by adding the Cloud Foundry subaccount X.509 certificate as trusted IdP in the Neo subaccount. Thus, the Cloud Foundry application starts acting as an IdP and any users propagated by it are accepted by the Neo application, even users that do not exist in the IdP.

The OAuth service accepts the SAML assertion and returns an OAuth access token. In turn, the Destination service returns both the destination and the access token to the requesting application. The application then uses the destination properties and the access token to consume the remote API.

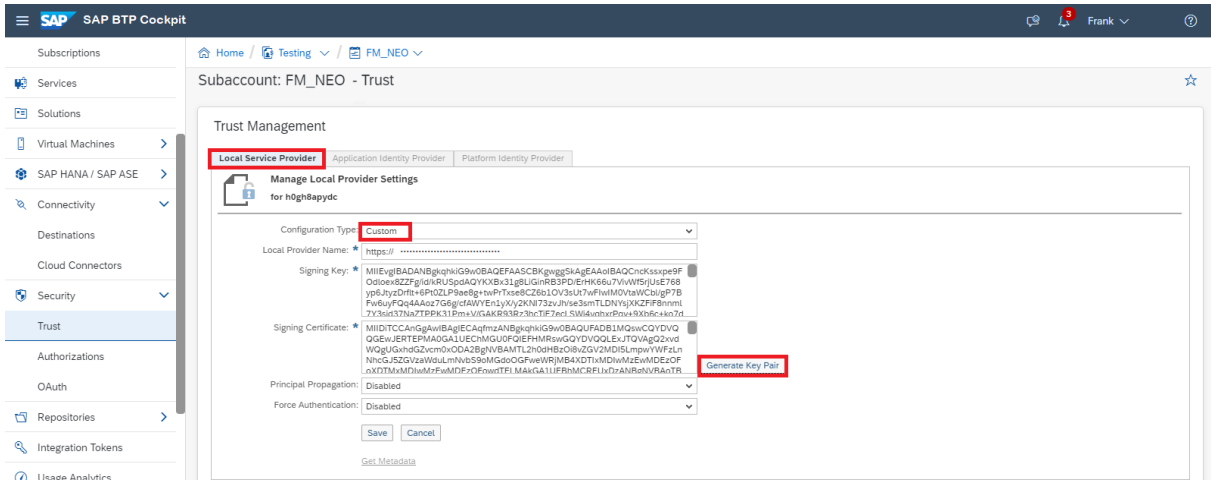


Back to [Steps \[page 196\]](#)

Procedure

Configure a Local Service Provider for the Neo Subaccount

1. In the cockpit, navigate to your Neo subaccount, choose **Security** > **Trust** from the left menu, and go to tab *Local Service Provider* on the right. For **Configuration Type**, select **Custom** and choose *Generate Key Pair*.
2. *Save* the configuration.



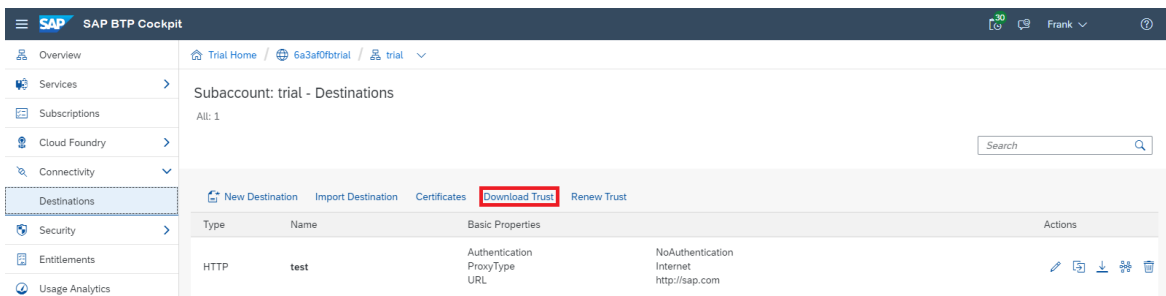
Note

IMPORTANT: When you choose **Custom** for the *Local Service Provider* type, the default IdP (SAP ID service) will no longer be available. If your scenario requires login to the SAP ID service as well, you can safely skip this step and leave the default settings for the Local Service Provider.

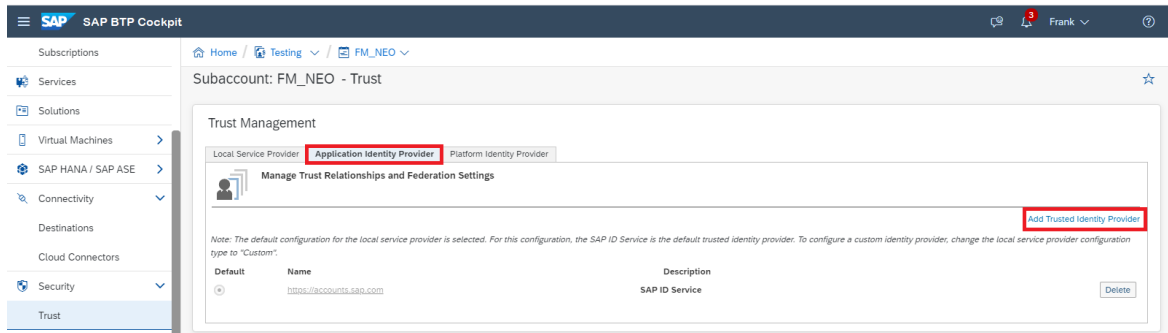
Back to [Steps \[page 196\]](#)

Establish Trust between Cloud Foundry and Neo Subaccounts

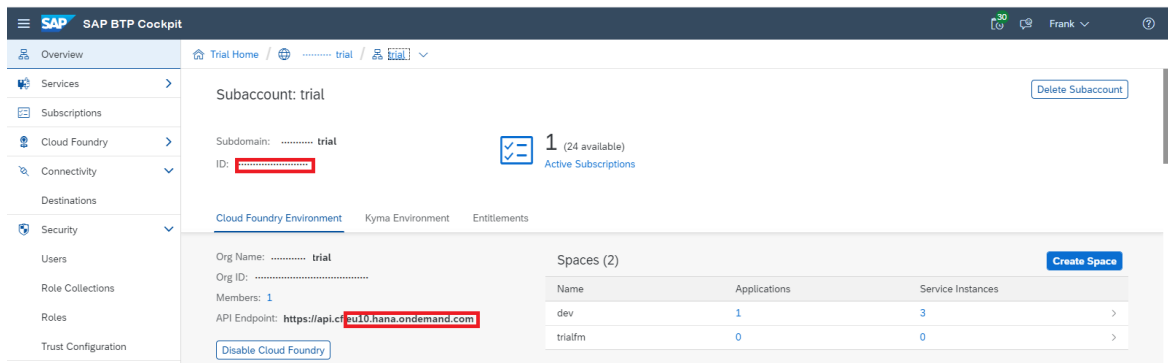
1. Download the X.509 certificate of the Cloud Foundry subaccount:
In the cockpit, navigate to your Cloud Foundry subaccount and choose **Connectivity** **Destinations**. Press **Download Trust** to get the certificate for this subaccount.



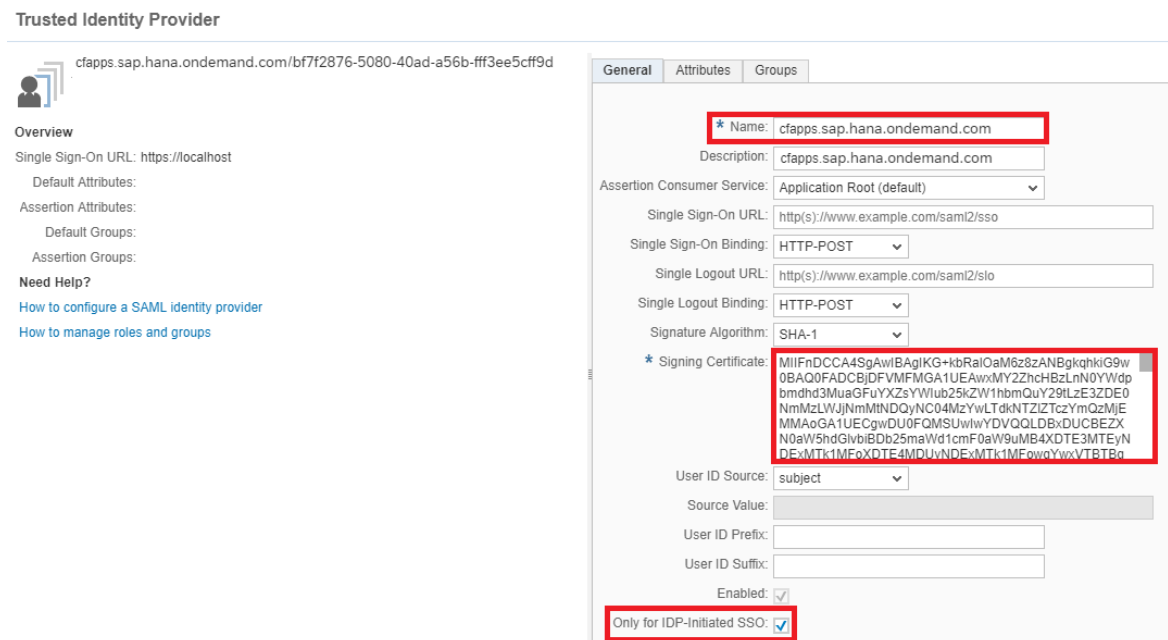
2. Configure trust in the Neo subaccount:
In the cockpit, navigate to your Neo subaccount, choose **Trust** from the left menu, and select the **Application Identity Provider** tab on the right. Then choose **Add Trusted Identity Provider**.



In the **<Name>** field, enter the *cfapps* host followed by the subaccount GUID, for example `cfapps.sap.hana.ondemand.com/bf7f2876-5080-40ad-a56b-fff3ee5cff9d`. This information is available in the cockpit, on the overview page of your Cloud Foundry subaccount:



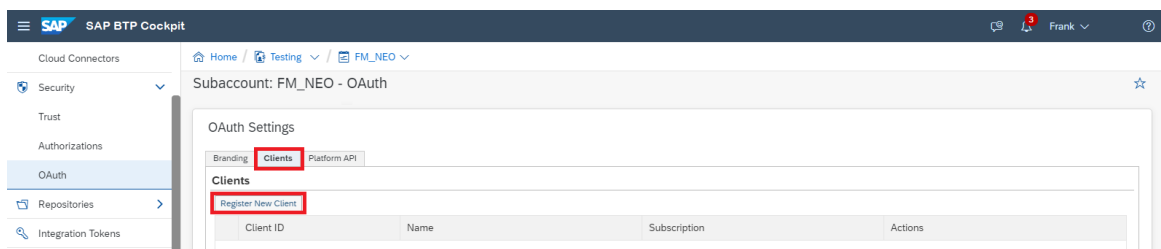
In the **<Signing Certificate>** field, paste the X.509 certificate you downloaded in step 1. Make sure you remove the *BEGIN CERTIFICATE* and *END CERTIFICATE* strings. Then check *Only for IDP-Initiated SSO* and save the configuration:



Back to [Steps \[page 196\]](#)

Create an OAuth Client for the Neo Application

1. In the cockpit, navigate to the Neo subaccount, choose **Security** > **OAuth** from the left menu, select tab **Client**, and choose **Register New Client**:

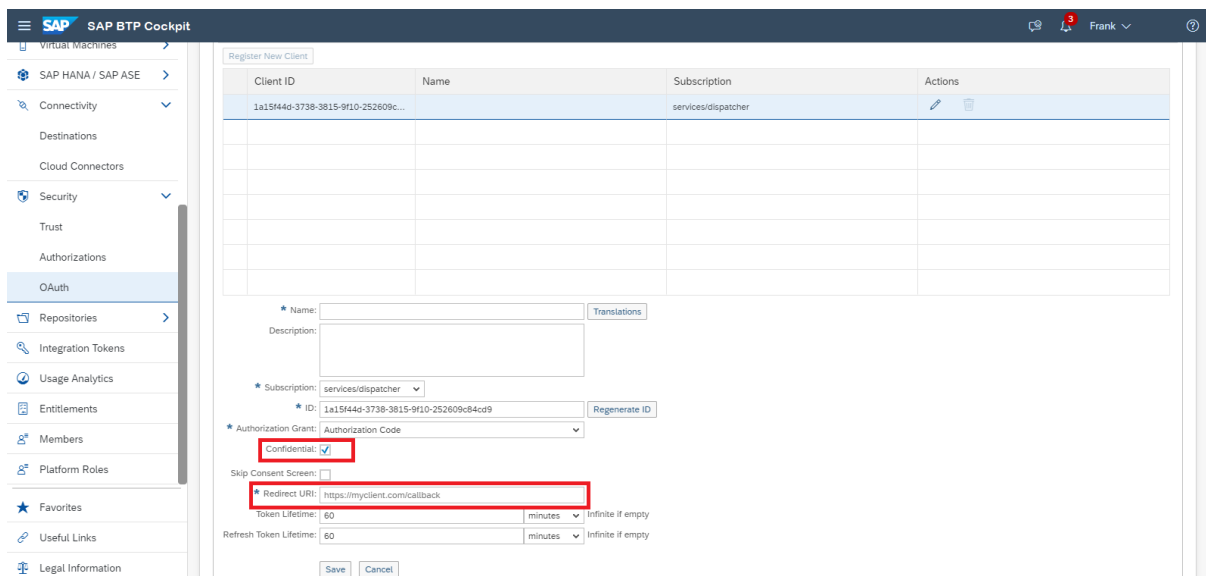


2. Enter a <Name> for the client.
3. In the <Subscription> field, select your Neo application.
4. For <Authorization Grant> select **Authorization Code**.
5. Check the **Confidential** checkbox and provide a secret for the OAuth client.

Note

Make sure you remember the secret, because it will not be visible later.

6. <Redirect URI> is irrelevant for the OAuth SAML Bearer Assertion flow, so you can provide any URL in the Cloud Foundry application.

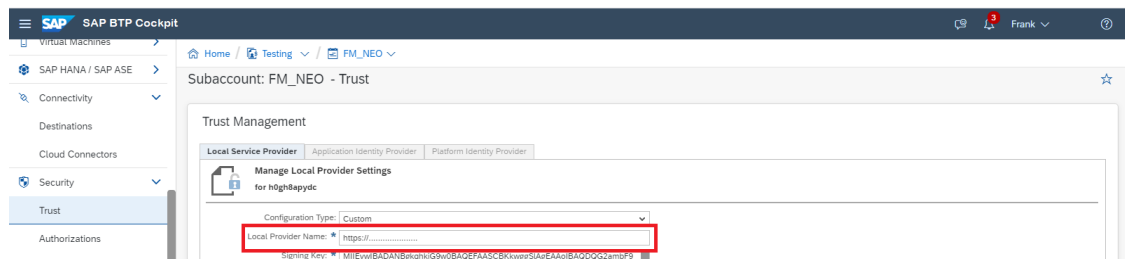


Back to [Steps \[page 196\]](#)

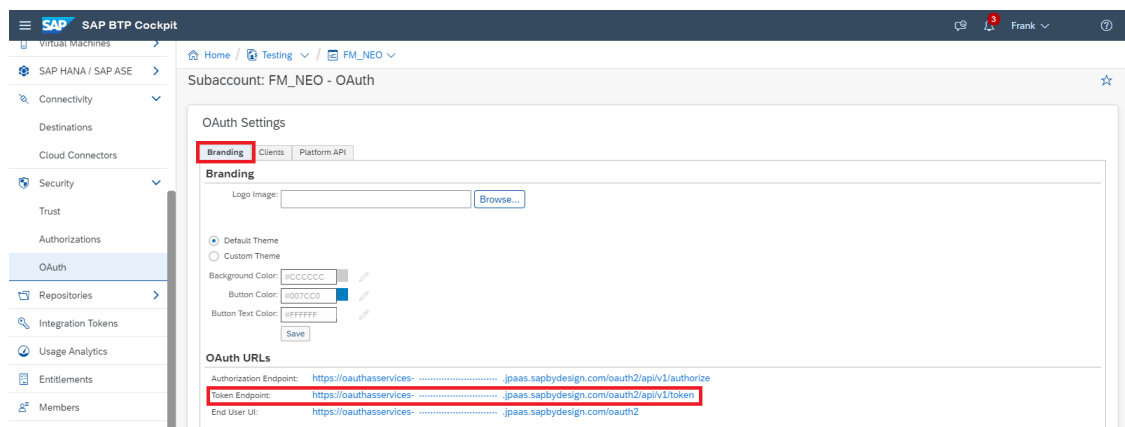
Create an OAuth2SAMLBearerAssertion Destination for the Cloud Foundry Application

1. In the cockpit, navigate to the Cloud Foundry subaccount, choose **Connectivity** > **Destinations** from the left menu, select the **Client** tab and press **New Destination**.
2. Enter a <Name> for the destination, then provide:
 - <URL>: the URL of the Neo application/API you want to consume.
 - <Authentication>: **OAuth2SAMLBearerAssertion**

- **<Audience>**: can be taken from the Neo subaccount, if you choose **Security > Trust** from the left menu, go to the *Local Service Provider* tab, and copy the value of **<Local Provider Name>**:



- **<Client Key>**: the ID of the OAuth client for the Neo application
- **<Token Service URL>**: can be taken from the *Branding* tab in the Neo subaccount (choose **Security > OAuth** from the left menu):



- **<Token Service User>**: again the ID of the OAuth client for the Neo application.
- **<Token Service Password>**: the OAuth client secret.

Enter two additional properties:

- **authnContextClassRef**: `urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession`
- **nameIdFormat**:
 - `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`, if the user ID is propagated to the Neo application, or
 - `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`, if the user email is propagated to the Neo application.

Destination Configuration

*Name:	neo	Additional Properties	<input type="button" value="New Propri"/>
Type:	HTTP	authnContextClassRef	urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSessio
Description:		nameIdFormat	urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddre
*URL:	https://neouserpropagationtrial.nana.ondemand.com	<input checked="" type="checkbox"/> Use default JDK truststore	
Proxy Type:	Internet		
Authentication:	OAuth2SAMLBearerAssertion		
*Audience:	https://hana.ondemand.com/trial		
*Client Key:	*****		
*Token Service URL:	https://oauthservices-trial.nana.ondemand.com/oauth2/api/		
Token Service User:	2afd27bf-4297-3faa-b1a2-bffae3a58fe8		
Token Service Password:	*****		
System User:			

Back to [Steps \[page 196\]](#)

Consume the Destination and Execute the Scenario

To perform the scenario and execute the request from the source application towards the target application, proceed as follows:

1. Decide on where the user identity will be located when calling the Destination service. For details, see [User Propagation via SAML 2.0 Bearer Assertion Flow \[page 251\]](#). This will determine how exactly you will perform step 2.
2. Execute a "find destination" request from the source application to the Destination service. For details, see [Consuming the Destination Service \[page 243\]](#) and the [REST API documentation](#) .
3. From the Destination service response, extract the access token and URL, and construct your request to the target application. See "[Find Destination](#)" [Response Structure \[page 259\]](#) for details on the structure of the response from the Destination service.

Back to [Steps \[page 196\]](#)

1.1.3.5 Multitenancy in the Connectivity Service

Using multitenancy for Cloud Foundry applications that require a connection to a remote service or on-premise application.

Endpoint Configuration

Applications that require a connection to a remote service can use the Connectivity service to configure HTTP or RFC endpoints. In a provider-managed application, such an endpoint can either be once defined by the application provider ([Provider-Specific Destination \[page 204\]](#)), or by each application subscriber ([Subscriber-Specific Destination \[page 205\]](#)).

If the application needs to use the same endpoint, independently from the current application subscriber, the destination that contains the endpoint configuration is uploaded by the application provider. If the endpoint should be different for each application subscriber, the destination can be uploaded by each particular application subscriber.

Note

This connectivity type is fully applicable also for on-demand to on-premise connectivity.

Destination Levels

You can configure destinations simultaneously on two levels: *subaccount* and *service instance*. This means that it is possible to have one and the same destination on more than one configuration level. For more information, see [Managing Destinations \[page 59\]](#).

Destination lookup according to the level, when configured on:

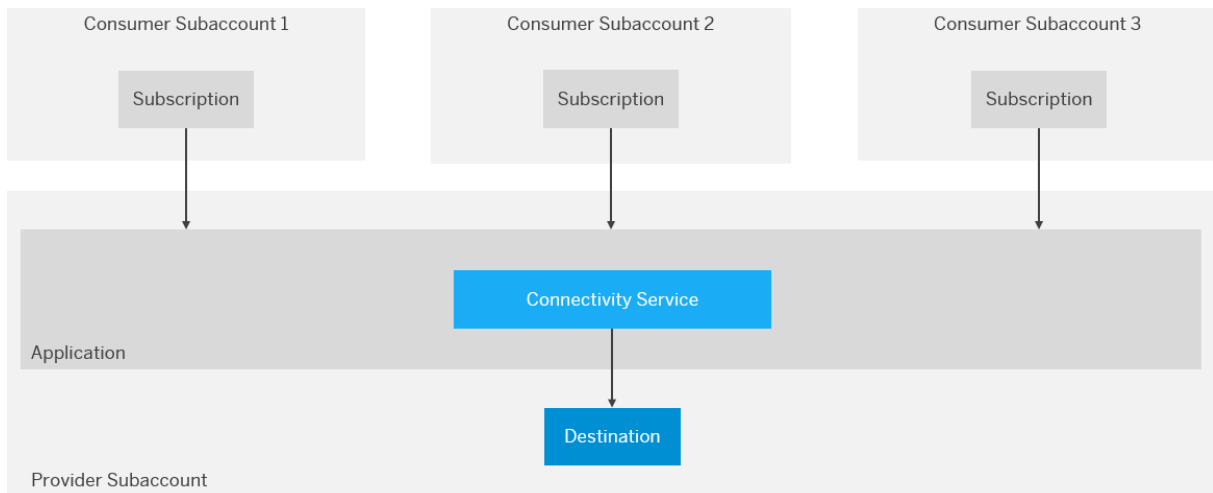
Level	Lookup
Subaccount level	Looked up on subaccount level, no matter which destination service instance is used.
Service instance level	Lookup via particular service instance (in provider or subscriber subaccount associated with this service instance).

When the application accesses the destination at runtime, the Connectivity service does the following:

- For a destination associated with a **provider** subaccount:
 1. Checks if the destination is available on the *service instance* level. If there is no destination found, it
 2. Searches the destination on *subaccount* level.
- For a destination associated with a **subscriber** subaccount:
 1. Checks if the destination is available on the *subscription* level. If there is no destination found, it
 2. Searches the destination on *subaccount* level.

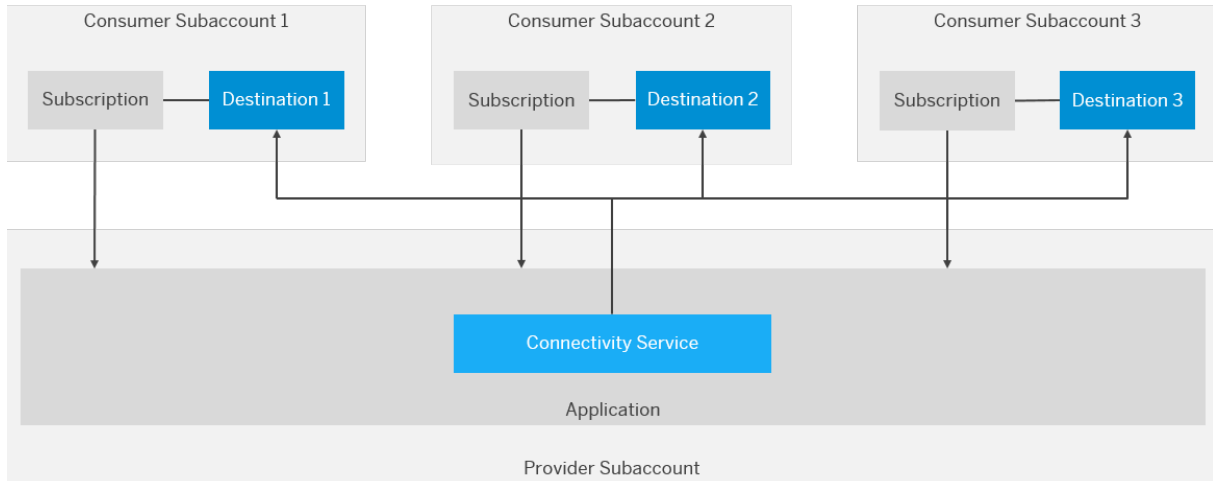
Back to [Top \[page 203\]](#)

Provider-Specific Destination



Back to [Top \[page 203\]](#)

Subscriber-Specific Destination



Back to [Top \[page 203\]](#)

Related Information

[Developing Multitenant Applications in the Cloud Foundry Environment](#)

1.1.3.6 Create and Bind a Connectivity Service Instance

To use the Connectivity service in your application, you need an instance of the service.

Prerequisites

When using service plan “lite”, quota management is no longer required for this service. From any subaccount you can consume the service using service instances without restrictions on the instance count.

Previously, access to service plan “lite” has been granted via entitlement and quota management of the application runtime. It has now become an integral service offering of SAP BTP to simplify its usage. See also [Entitlements and Quotas](#).

Procedure

You have two options for creating a service instance – using the CLI or using the SAP BTP cockpit:

- [Create and Bind a Service Instance from the CLI \[page 206\]](#)
 - [Example \[page 207\]](#)
 - [Result \[page 208\]](#)
- [Create and Bind a Service Instance from the Cockpit \[page 207\]](#)
 - [Result \[page 208\]](#)

For more information on using X.509 certificates, see:

[X.509 Bindings \[page 208\]](#)

Create and Bind a Service Instance from the CLI

Use the following CLI commands to create a service instance and bind it to an application:

1. `cf marketplace`
2. `cf create-service connectivity <service-plan> <service-name>`
3. `cf bind-service <app-name> <service-name> [-c <config json>]`

Back to [Procedure \[page 206\]](#)

Example

To bind an instance of the Connectivity service "lite" plan to application "myapp", use following commands on the Cloud Foundry command line:

```
cf create-service connectivity lite myinstance
```

```
cf bind-service myapp myinstance
```

Back to [Procedure \[page 206\]](#)

Create and Bind a Service Instance from the Cockpit

Assuming that you have already deployed your application to the platform, follow these steps to create a service instance and bind it to an application:

1. In the SAP BTP Cockpit, navigate to your application.
2. From the left navigation menu, choose *Service Bindings*.
3. Choose the *Bind Service* button.
4. The *Bind Service* wizard appears.
5. Select the *Service from the catalog* radio button, then choose *Next*.
6. From the list of available services, select *Connectivity*, then choose *Next*.

Bind Service

Choose Service Type

Choose Service

Choose Service

Choose Service Plan

Specify Parameters

Confirm

Description: Establishes a secure and reliable connectivity between cloud applications and on-premise systems.

blockchain-services

business-entity-reco...

dataenrichment-busi...

Business Rules business-rules

Cloud Management cis

Connectivity connectivity

Content Agent content-agent

Credential Store credstore

Data Attribute Recommendation

Previous Next Cancel

7. On the next page of the wizard, select the *Create new instance* radio button.
8. Leave *<Plan> lite* selected and choose *Next*.

9. The next page is used for specifying user-provided parameters in JSON format. If you do not want to do that, skip this step by choosing [Next](#).
10. In the `<Instance Name>` textbox, enter an unique name for your service instance.
11. Choose [Finish](#).

Back to [Procedure \[page 206\]](#)

Result

When the binding is created, the application gets the corresponding connectivity credentials in its environment variables:

Sample Code

```
"VCAP_SERVICES": {
  "connectivity": [
    {
      "credentials": {
        "onpremise_proxy_host": "10.0.85.1",
        "onpremise_proxy_port": "20003",
        "onpremise_proxy_http_port": "20003",
        "clientid": "sb-connectivity-app",
        "clientsecret": "KXqObiN6d9gLA4cS2rOVAahPCX0=",
        "token_service_url": "<token_service_url>",
      },
      "label": "connectivity",
      "name": "conn-lite",
      "plan": "default",
      "provider": null,
      "syslog_drain_url": null,
      "tags": [
        "connectivity",
        "conn",
        "connsvc"
      ],
      "volume_mounts": []
    }
  ],
}
```

Note

"onpremise_proxy_http_port" replaces the deprecated variable "onpremise_proxy_port", which will be removed soon. Same goes for "token_service_url", which replaces "url".

Back to [Procedure \[page 206\]](#)

X.509 Bindings

Note

Older instances of the service do not support the X.509 credentials binding type and an attempt to create one will result in an error. To overcome this, you just need to update the service instance, so it picks up the X.509 support.

The service supports X.509 bindings as described in [Retrieving Access Tokens with Mutual Transport Layer Security \(mTLS\)](#). This lets you choose if your binding / service key will be with a client secret X.509 certificate generated by SAP or an X.509 certificate provided by you. To do so, add a config JSON when creating your binding:

```
cf bind-service <app-name> <service-name> [-c <config json>]
```

The structure of the config JSON should be like this:

```
{
  "xsuaa":
  {
    <X.509 properties>
  },
  <other config parameters>
}
```

For a list of the supported X.509 properties, see:

- [Parameters for X.509 Certificates Managed by SAP Authorization and Trust Management Service](#)
- [Parameters for Self-Managed X.509 Certificates](#)

Note

By default, without providing a config JSON, a binding with client secret will be created.

Back to [Procedure \[page 206\]](#)

1.1.3.7 Create and Bind a Destination Service Instance

To use the Destination service in your application, you need an instance of the service.

Concept

To consume the Destination service, you must provide the appropriate credentials through a service instance and a service binding/service key. The Destination service is publicly visible and cross-consumable from

several environments and provides the service plan *lite* to all those environments. Provisioning a service instance and service key is done in the standard way for the respective environment, see:

- Cloud Foundry: [Using Services in the Cloud Foundry Environment](#)
- Kyma: [Using SAP BTP Services in the Kyma Environment](#)
- Kubernetes: [Consuming SAP BTP Services in Kubernetes with SAP Service Manager Broker Proxy \(Service Catalog\)](#)
- Other environments: [Consuming Services in Other Environments Using the SAP Service Manager Instances](#)

In all environments, the Destination service lets you provide a configuration JSON during instance creation or update.

Using a Configuration JSON

You can pass a configuration JSON during instance creation or update to modify some of the default settings of the instance and/or provide some content to be created during the operation.

The detailed structure of the configuration JSON is described in [Use a Config.JSON to Create or Update a Destination Service Instance \[page 211\]](#).

Troubleshooting

If you get this failure message:

```
Failed to create all provided configurations. Will delete all on instance level, for configurations on subaccount level you can set policies to handle duplicates: "existing_destinations_policy" with value "update|fail|ignore" and "existing_certificates_policy" with value "fail|ignore".
```

the root cause may be:

- A provided destination or certificate with the same name already exists in this subaccount. Solution: set policies on subaccount level to update or ignore in case of conflicts.

```
"existing_destinations_policy": "update|fail|ignore"  
"existing_certificates_policy": "fail|ignore"
```

- A client input error occurred. Solution: check the input, apply correction and try again.
- An internal server error occurred. In this case, please try again later or report a support incident.

X.509 Bindings

Note

Older instances of the service do not support the X.509 credentials binding type and an attempt to create one will result in an error. To overcome this, you just need to update the service instance, so it picks up the X.509 support.

The service supports X.509 bindings as described in [Retrieving Access Tokens with Mutual Transport Layer Security \(mTLS\)](#). This lets you choose if your binding / service key will be with a client secret X.509 certificate generated by SAP or an X.509 certificate provided by you. To do so, add a config JSON when creating your binding:

```
cf bind-service <app-name> <service-name> [-c <config json>]
```

The structure of the config JSON should be like this:

```
{
  "xsuaa":
  {
    <X.509 properties>
  },
  <other config parameters>
}
```

For a list of the supported X.509 properties, see:

- [Parameters for X.509 Certificates Managed by SAP Authorization and Trust Management Service](#)
- [Parameters for Self-Managed X.509 Certificates](#)

Note

By default, without providing a config JSON, a binding with client secret will be created.

1.1.3.7.1 Use a Config.JSON to Create or Update a Destination Service Instance

Configure specific parameters in a `config.json` file to create or update a Destination service instance.

When creating or updating a Destination service instance, you can configure the following settings, which are part of the `config.json` input file (see [Open Service Broker API](#)), both via SAP BTP cockpit or Cloud Foundry command line interface (CLI):

Parameter	Value	Description
init_data	JSON	The data (destinations, certificates) to initialise or update the service instance with. The data can be stored on both <i>service instance</i> data and <i>subaccount</i> data.
HTML5Runtime_enabled	Boolean	Indicates whether the SAP BTP HTML5 runtime should be enabled to work with the service instance on behalf of the HTML5 applications associated with it during deployment.

Find the `config.json` structure below:

Sample Code

```
{
  "HTML5Runtime_enabled" : true,
  "init_data" : {
    "subaccount" : {
      "existing_destinations_policy": "update|fail|ignore",
      "existing_certificates_policy": "update|fail|ignore",
      "destinations" : [
        {
          ...
        }
      ],
      "certificates" : [
        {
          ...
        }
      ]
    },
    "instance" : {
      "existing_destinations_policy": "update|fail|ignore",
      "existing_certificates_policy": "update|fail|ignore",
      "destinations" : [
        {
          ...
        }
      ],
      "certificates" : [
        {
          ...
        }
      ]
    }
  }
}
```

Related Information

[HTTP Destinations \[page 86\]](#)

1.1.3.8 Configuring Backup

To create a backup of your destination configurations, choose the procedure [Export Destinations \[page 80\]](#).

1.1.3.9 Notifications and Alerts

Subscribe to the SAP Alert Notification service to receive notifications and alerts for the Destination service on SAP BTP, Cloud Foundry environment.

You can subscribe for notifications and alerts for different Destination service events via SAP Alert Notification Service for SAP BTP.

For more information, see [What Is SAP Alert Notification Service for SAP BTP?](#)

For details on Destination service events, see [SAP BTP Destination Service Events](#).

1.1.3.10 Destination Fragments

Destination fragments are objects used to override and extend destination properties through the Destination service REST API.

You can use destination fragments to override and/or extend destination properties as result of the "Find a destination" REST API request. Destination fragments are key-value based objects which contain a name and additional configurable properties.

Property	Description
FragmentName	Name of the fragment. Must be unique for the level on which it is stored/maintained.

⚠ Restriction

The fragment must not contain the properties "Name" or "Type".

Managing destination fragments for your application is supported only by the Destination service REST API. This API is documented in the [SAP Business Accelerator Hub](#).

Related Information

[Extending Destinations with Fragments \[page 273\]](#)

[Calling the Destination Service REST API \[page 256\]](#)

1.1.4 Developing Applications

Consume the Connectivity service and the Destination service from an application in the Cloud Foundry environment.

Task	Description
Consuming the Connectivity Service [page 214]	Connect your Cloud Foundry application to an on-premise system.
Consuming the Destination Service [page 243]	Retrieve and store externalized technical information about the destination that is required to consume a target remote service from your application.
Invoking ABAP Function Modules via RFC [page 277]	Call a remote-enabled function module (RFM) in an on-premise or cloud ABAP server from your Cloud Foundry application, using the RFC protocol.

1.1.4.1 Consuming the Connectivity Service


Connect your Cloud Foundry application to an on-premise system via HTTP.



Note

To use the Connectivity service with a protocol other than HTTP, see

- [Invoking ABAP Function Modules via RFC \[page 277\]](#)
- [Using the TCP Protocol for Cloud Applications \[page 234\]](#)

Tasks

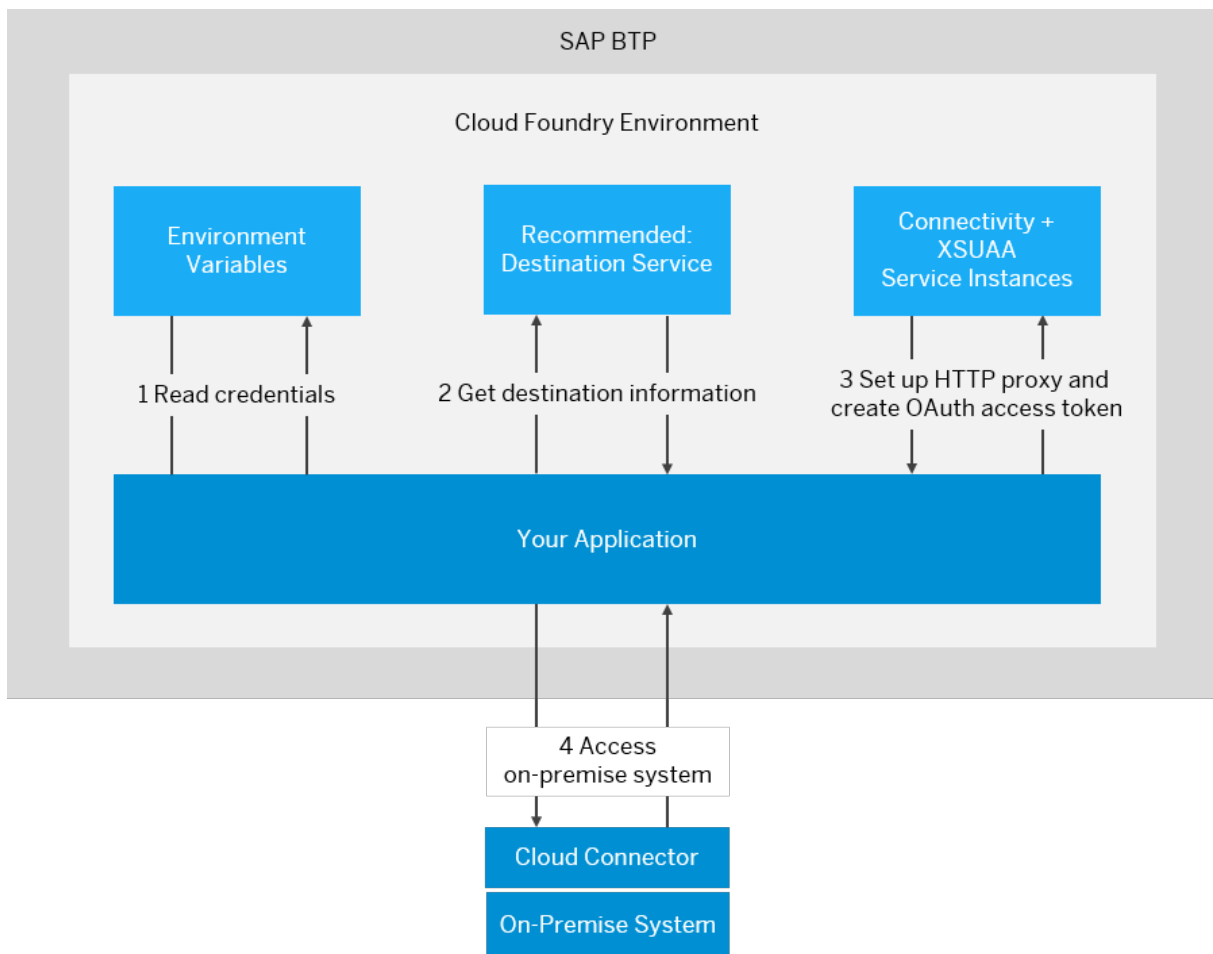
Task Type	Task
	Overview [page 215]
	Prerequisites [page 216]
Operator and/or Developer	

Task Type	Task
 Developer	<p data-bbox="804 353 1027 383">Basic Steps [page 217]</p> <ol data-bbox="804 405 1394 577" style="list-style-type: none"> <li data-bbox="804 405 1394 472">1. Read Credentials from the Environment Variables [page 218] <li data-bbox="804 479 1394 508">2. Provide the Destination Information [page 218] <li data-bbox="804 515 1394 577">3. Set up the HTTP Proxy for On-Premise Connectivity [page 219]
 Operator and/or Developer	<p data-bbox="804 616 1075 645">Additional Steps [page 221]</p> <ul data-bbox="804 667 1394 817" style="list-style-type: none"> <li data-bbox="804 667 1394 734">• Authentication against the On-Premise System [page 221] <li data-bbox="804 741 1394 770">• Specify a Cloud Connector Location ID [page 221] <li data-bbox="804 777 1394 817">• Multitenancy in the Connectivity Service [page 222]

Overview



Using the Connectivity service, you can connect your Cloud Foundry application to an on-premise system through the Cloud Connector. To achieve this, you must provide the required information about the target system (destination), and set up an HTTP proxy that lets your application access the on-premise system.



Back to [Tasks \[page 214\]](#)

Prerequisites



- You must be a *Global Account* member to connect through the Connectivity service with the Cloud Connector. See [Add Members to Your Global Account](#). Also *Security Administrators* (which must be either Global Account members or Cloud Foundry Organization/Space members) can do it. See [Managing Security Administrators in Your Subaccount \[Feature Set A\]](#).

Note

To connect a Cloud Connector to your subaccount, you must currently be a **Security Administrator**.

- You have installed and configured a Cloud Connector in your on-premise landscape for to the scenario you want to use. See [Installation \[page 349\]](#) and [Configuration \[page 387\]](#).

- You have deployed an application in a landscape of the Cloud Foundry environment that complies with the [Business Application Pattern](#).
- The Connectivity service is a regular service in the Cloud Foundry environment. Therefore, to consume the Connectivity service from an application, you must create a service instance and bind it to the application. See [Create and Bind a Connectivity Service Instance \[page 206\]](#).
- To get the required authorization for making on-premise calls through the connected Cloud Connector, the application must be bound to an instance of the `xsuaa` service using the service plan 'application'. The `xsuaa` service instance acts as an OAuth 2.0 client and grants user access to the bound application. Make sure you set the `xsappid` property to the name of the application when creating the instance. Find a detailed guide for this procedure in section 3. *Creation of the Authorization & Trust Management Instance (aka XSUAA)* of the SCN blog [How to use SAP BTP Connectivity and Cloud Connector in the Cloud Foundry environment](#).

Note

Currently, the only supported protocol for connecting the Cloud Foundry environment to an on-premise system is HTTP. HTTPS is not needed, since the tunnel used by the Cloud Connector is TLS-encrypted.

Caution

There is a limit of **8192** bytes for the size of the HTTP lines (for example, request line or header) that you send via the Connectivity service. If this limit is exceeded, you receive an HTTP error of type 4xx. This issue is usually caused by the size of the `path + query` string of the request.

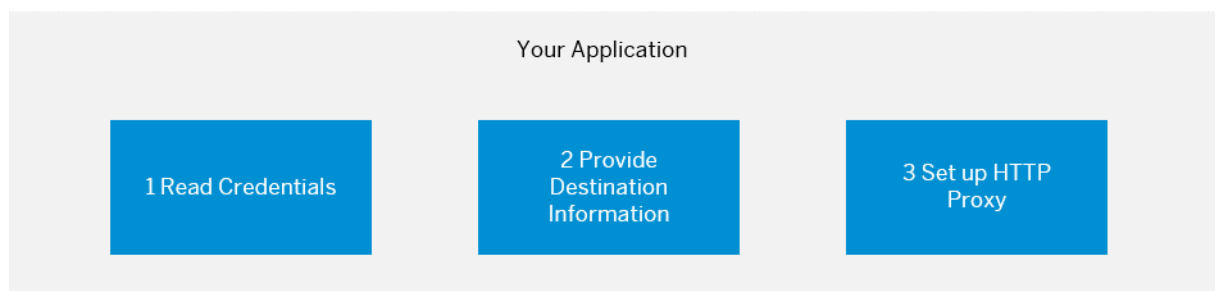
Back to [Tasks \[page 214\]](#)

Basic Steps



To consume the Connectivity service from your Cloud Foundry application, perform the following basic steps:

1. [Read Credentials from the Environment Variables \[page 218\]](#)
2. [Provide the Destination Information \[page 218\]](#)
3. [Set up the HTTP Proxy for On-Premise Connectivity \[page 219\]](#)



Back to [Tasks \[page 214\]](#)

Read Credentials from the Environment Variables



Consuming the Connectivity service requires credentials from the *xsuaa* and *Connectivity* service instances which are bound to the application. By binding the application to service instances of the *xsuaa* and *Connectivity* service as described in the prerequisites, these credentials become part of the environment variables of the application. You can access them as follows:

Sample Code

```
JSONObject jsonObj = new JSONObject(System.getenv("VCAP_SERVICES"));
JSONArray jsonArr = jsonObj.getJSONArray("<service name, not the instance name>");
JSONObject credentials =
    jsonArr.getJSONObject(0).getJSONObject("credentials");
```

Note

If you have multiple instances of the same service bound to the application, you must perform additional filtering to extract the correct credential from `jsonArr`. You must go through the elements of `jsonArr` and find the one matching the correct instance name.

This code stores a JSON object in the `credentials` variable. Additional parsing is required to extract the value for a specific key.

Note

We refer to the result of the above code block as `connectivityCredentials`, when called for *connectivity*, and `xsuaaCredentials` for *xsuaa*.

Back to [Tasks \[page 214\]](#)

Provide the Destination Information



To consume the Connectivity service, you must provide some information about your on-premise system and the system mappings for it in the Cloud Connector. You require the following:

- The endpoint in the Cloud Connector (virtual host and virtual port) and accessible URL paths on it (destinations). See [Configure Access Control \(HTTP\) \[page 457\]](#).
- The required authentication type for the on-premise system. See [HTTP Destinations \[page 86\]](#).
- Depending on the authentication type, you may need a username and password for accessing the on-premise system. For more details, see [Client Authentication Types for HTTP Destinations \[page 101\]](#).
- (Optional) You can use a *location Id*. For more details, see section [Specify a Cloud Connector Location ID \[page 221\]](#).

We recommend that you use the *Destination service* (see [Consuming the Destination Service \[page 243\]](#)) to procure this information. However, using the Destination service is optional. You can also provide (look up) this information in another appropriate way.

Back to [Tasks \[page 214\]](#)

Set up the HTTP Proxy for On-Premise Connectivity



Proxy Setup

The Connectivity service provides a standard HTTP proxy for on-premise connectivity that is accessible by any application. Proxy host and port are available as the environment variables `<onpremise_proxy_host>` and `<onpremise_proxy_http_port>`. You can set up the on-premise HTTP proxy like this:

Sample Code

```
// get value of "onpremise_proxy_host" and "onpremise_proxy_http_port" from
the environment variables
// and create on-premise HTTP proxy
String connProxyHost =
connectivityCredentials.getString("onpremise_proxy_host");
int connProxyPort =
Integer.parseInt(credentials.getString("onpremise_proxy_http_port"));
Proxy proxy = new Proxy(Proxy.Type.HTTP, new InetSocketAddress(connProxyHost,
connProxyPort));

// create URL to the remote endpoint you like to call:
// virtualhost:1234 is defined as an endpoint in the Cloud Connector, as
described in the Required Information section
URL url = new URL("http://virtualhost:1234");

// create the connection object to the endpoint using the proxy
// this does not open a connection but only creates a connection object,
which can be modified later, before actually connecting
URLConnection = (HttpURLConnection) url.openConnection(proxy);
```

Note

"onpremise_proxy_http_port" replaces the deprecated variable "onpremise_proxy_port", which will be removed soon.

Authorization

To make calls to on-premise services configured in the Cloud Connector through the HTTP proxy, you must authorize at the HTTP proxy. For this, the *OAuth Client Credentials* flow is used: applications must create an OAuth access token using the parameters `clientId` and `clientsecret` that are provided by the Connectivity service in the environment, as shown in the example code below. When the application has retrieved the access token, it must pass the token to the connectivity proxy using the `Proxy-Authorization` header.

The sample code below uses the following Maven artifacts:

- `org.springframework.security:spring-security-oauth2-core`
- `org.springframework.security:spring-security-oauth2-client`

Sample Code

```
import
org.springframework.security.authentication.AbstractAuthenticationToken;
import
org.springframework.security.oauth2.client.ClientCredentialsOAuth2AuthorizedCl
ientProvider;
import org.springframework.security.oauth2.client.OAuth2AuthorizationContext;
import
org.springframework.security.oauth2.client.OAuth2AuthorizedClientProvider;
import
org.springframework.security.oauth2.client.registration.ClientRegistration;
import org.springframework.security.oauth2.core.AuthorizationGrantType;
import org.springframework.security.oauth2.core.OAuth2AccessToken;

...

// get value of "clientId" and "clientsecret" from the environment variables
String clientId = connectivityCredentials.getString("clientId");
String clientsecret = connectivityCredentials.getString("clientsecret");

// get the URL to xsuaa from the environment variables
String xsuaaUrl = xsuaaCredentials.getString("token_service_url");

// make request to UAA to retrieve access token
ClientRegistration clientRegistration =
ClientRegistration.withRegistrationId("some-id").
authorizationGrantType(AuthorizationGrantType.CLIENT_CREDENTIALS).
clientId(clientId).
clientSecret(clientsecret).
authorizationUri(xsuaaUrl + "/oauth/authorize").
tokenUri(xsuaaUrl + "/oauth/token").
build();

OAuth2AuthorizationContext xsuaaContext =
OAuth2AuthorizationContext.withClientRegistration(clientRegistration).
principal(new AbstractAuthenticationToken(null) {
@Override
public Object getPrincipal() {
return null;
}
```

```

    }

    @Override
    public Object getCredentials() {
        return null;
    }

    @Override
    public String getName() {
        return "dummyPrincipalName"; // There is no principal in the
client credentials authorization grant but a non-empty name is still required.
    }
}).build();

OAuth2AuthorizedClientProvider clientCredentialsAccessTokenProvider = new
ClientCredentialsOAuth2AuthorizedClientProvider();
OAuth2AccessToken token =
clientCredentialsAccessTokenProvider.authorize(xsuaaContext).getAccessToken();

// set access token as Proxy-Authorization header in the URL connection
urlConnection.setRequestProperty("Proxy-Authorization",
token.getTokenType().getValue() + " " + token.getTokenValue());

```

Note

`xsuaaCredentials.getString("token_service_url")` replaces the deprecated property `xsuaaCredentials.getString("url")`, which will be removed soon.

Back to [Tasks \[page 214\]](#)

Authentication against the On-Premise System



Depending on the required authentication type for the desired on-premise resource, you may have to set an additional header in your request. This header provides the required information for the authentication process against the on-premise resource. See [Authentication to the On-Premise System \[page 223\]](#).

Back to [Tasks \[page 214\]](#)

Specify a Cloud Connector Location ID



Note

This is an advanced option when using more than one Cloud Connector for a subaccount. For more information how to set the `location ID` in the Cloud Connector, see [Managing Subaccounts \[page 401\]](#), step 4 in section *Subaccount Dashboard*.

You can connect multiple Cloud Connectors to a subaccount if their `location ID` is different. Using the header `SAP-Connectivity-SCC-Location_ID` you can specify the Cloud Connector over which the connection should be opened. If this header is not specified, the connection is opened to the Cloud Connector that is connected without any `location ID`. For example:

Sample Code

```
// Optionally, if configured, add the SCC location ID.  
urlConnection.setRequestProperty("SAP-Connectivity-SCC-Location_ID",  
"orlando");
```

Back to [Tasks \[page 214\]](#)

Multitenancy in the Connectivity Service



To consume the Connectivity service from an SaaS application in a multitenant way, the only requirement is that the SaaS application returns the Connectivity service as a dependent service in its dependencies list.

For more information about the subscription flow, see [Develop the Multitenant Business Application](#).

Back to [Tasks \[page 214\]](#)

Related Information

[Cloud Connector \[page 343\]](#)

[Set Up an Application as a Sample Backend System](#)

[Create and Bind a Connectivity Service Instance \[page 206\]](#)

[Authentication to the On-Premise System \[page 223\]](#)

[Consuming the Destination Service \[page 243\]](#)

[What Is the SAP Authorization and Trust Management Service?](#)

[Multitarget Applications in the Cloud Foundry Environment](#)

[Invoking ABAP Function Modules via RFC \[page 277\]](#)

[Using the TCP Protocol for Cloud Applications \[page 234\]](#)

1.1.4.1.1 Authentication to the On-Premise System

Provide authentication information for the authentication type you use.

You can use the Connectivity service in different authentication scenarios:

- **No authentication** to the on-premise system
- **Principal propagation** (user propagation) to the on-premise system
- **Basic authentication** to the on-premise system

Procedure

For each authentication type, you must provide specific information in the request to the virtual host:

- [The SAP-Connectivity-Authentication Header \[page 223\]](#)
- [Authentication Types \[page 224\]](#)

The SAP-Connectivity-Authentication Header

Note

For the principal propagation scenario, the `SAP-Connectivity-Authentication` header is only required if you do not use the user exchange token flow, see [Configure Principal Propagation via User Exchange Token \[page 227\]](#).

Applications must propagate the user JWT token (`userToken`) using the `SAP-Connectivity-Authentication` header. This is required for the Connectivity service to open a tunnel to the subaccount for which a configuration is made in the Cloud Connector. The following example shows you how to do this using the *Spring* framework:

Sample Code

```
Authentication auth = SecurityContextHolder.getContext().getAuthentication();
if (auth == null) {
    throw new UserNotFoundException("User not authenticated");
}
OAuth2AuthenticationDetails details = (OAuth2AuthenticationDetails)
auth.getDetails();
String userToken = details.getTokenValue();
urlConnection.setRequestProperty("SAP-Connectivity-Authentication", "Bearer "
+ userToken);
```

Back to [Procedure \[page 223\]](#)

Authentication Types

The required setup for each of the authentication type is as follows:

No Authentication

If the on-premise system does not need to identify the user, you should use this authentication type. It requires no additional information to be passed with the request.

Principal Propagation

When you open the application router to access your cloud application, you are prompted to log in. Doing so means that the cloud application now knows your identity. Principal propagation forwards this identity via the Cloud Connector to the on-premise system. This information is then used to grant access without additional input from the user. To achieve this, you do not need to send any additional information from your application, but you must set up the Cloud Connector for principal propagation. See [Configuring Principal Propagation \[page 420\]](#).

Note

Do not use the `Authorization` header in *principal propagation* scenarios.

Technical User Propagation

As of Cloud Connector version 2.15, consumers of the Connectivity service can propagate technical users from the cloud application towards the on-premise systems. To make use of this feature, provide a JWT (usually obtained via `client_credentials` OAuth flow), representing the technical user, via the `SAP-Connectivity-Technical-Authentication` header. This is similar to *principal propagation*, but in this case, a technical user is propagated instead of a business user.

```
urlConnection.setRequestProperty("SAP-Connectivity-Technical-Authentication",  
"Bearer " + technicalUserToken);
```

For more information, see [Configuring Principal Propagation \[page 420\]](#).

Note

Do not use the `Authorization` header in *technical user propagation* scenarios.

Basic Authentication

If the on-premise system requires username and password to grant access, the cloud application must provide these data using the `Authorization` header. The following example shows how to do this:

Sample Code

```
// Basic authentication to backend system  
String credentials = MessageFormat.format("{0}:{1}", backendUser,  
backendPassword);  
byte[] encodedBytes = Base64.encodeBase64(credentials.getBytes());
```

```
urlConnection.setRequestProperty("Authorization", "Basic " + new  
String(encodedBytes));
```

[Back to Procedure \[page 223\]](#)

Related Information

[Configure Principal Propagation via Corporate IdP Embedded Token \[page 225\]](#)

[Configure Principal Propagation via User Exchange Token \[page 227\]](#)




[Configure Principal Propagation via IAS Token \[page 231\]](#)

[Configure Principal Propagation via OIDC Token \[page 232\]](#)

1.1.4.1.1.1 Configure Principal Propagation via Corporate IdP Embedded Token

Configure a corporate IdP embedded token for principal propagation (user propagation) from your Cloud Foundry application to an on-premise system.

Tasks

Task Type	Task
 Operator and/or Developer	Scenario [page 226]
 Operator	Prerequisites [page 226]
 Developer	Solutions [page 226]

Scenario

For a Cloud Foundry application that uses the Connectivity service, you want the currently logged-in user to be propagated to an on-premise system via token from your trusted corporate IdP.

For more information, see [Principal Propagation \[page 168\]](#).

Back to [Tasks \[page 225\]](#)

Prerequisites

- Cloud Connector 2.13 (or newer) must be used.
- The Cloud Connector must be connected to a subaccount that is configured with the corporate IdP, issuing the tokens.

After the configuration, the issued XSUAA token contains an embedded token, which is extracted and propagated by the Connectivity service.

For more information, see [SAP Authorization and Trust Management Service in the Cloud Foundry Environment](#).

- Trust configuration for that subaccount must be synchronized in the Cloud Connector to obtain the JSON web key set from the configured corporate IdP that is used to verify the token.

For more information, see [Set Up Trust \[page 420\]](#).

Select Subaccount ⚙️ xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Cloud To On-Premise

ACCESS CONTROL COOKIE DOMAINS **PRINCIPAL PROPAGATION**

Trust Configuration (3)

Name	Description	Type	Trusted	Actions
Subaccount: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx ; kid: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	JSON Web Token (JWT) key, used by the SAP UAA (XSUAA) to sign JWT access tokens. Multiple keys might be configured for the same identity zone of the subaccount. Current key hash (sha1sum): 72CE5D087FD826E0C4C2A91DC7F968EBA5ED44B241865F9A852885169031E3E	IDP	✔️	🔗
Subaccount: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx ; kid: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	JSON Web Token (JWT) key, used by the SAP IAS to sign JWT access tokens. Multiple keys might be configured for the same IAS tenant, mapped to the subaccount. Current key hash (sha1sum): 0E42A804E9C41D978B17B88C64D732C9E0F6980D1F3CFE26169CE76EA56618E4	IDP	✔️	🔗
Subaccount: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx ; kid: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	JSON Web Token (JWT) key, used by the configured corporate IdP to sign JWT access tokens. Multiple keys might be configured for the same corporate IdP, mapped to the subaccount. Current key hash (sha1sum): DA5D441E29FF85AD3E34664D4F64D7AF187F5D3DDC385AABFB361FE174E8C.	IDP	✔️	🔗

Back to [Tasks \[page 225\]](#)

Solutions

You have two options to implement the user propagation via embedded corporate IdP token:

1. **Recommended.** The application sends one header containing the user exchange token to the Connectivity proxy:
 - The application sends one HTTP header `Proxy-Authorization`.

- The original *user token* is exchanged for a special *user exchange access token* following the OAuth2 JWT Bearer grant type.
For more information, see [Using JWTs as Authorization Grants](#) .
- The *user exchange access token* is send through the `Proxy-Authorization` header to consume the Connectivity service.
- The embedded corporate token which contains the user details must be present in the obtained *user exchange access token*.

```
Header: "Proxy-Authorization" : "Bearer <userExchangeAccessToken>"
```

For more information on how to obtain a user exchange access token with an embedded IAS corporate IdP token, see [SAP Authorization and Trust Management Service in the Cloud Foundry Environment](#) and [Generate the Authentication Token \[page 229\]](#).

2. The application sends two headers to the Connectivity proxy:
 - The application sends two HTTP headers: `SAP-Connectivity-Authentication` and `Proxy-Authorization`.
 - The embedded corporate token which contains the user details must be present in the user token provided via `SAP-Connectivity-Authentication`.
 - The access token is provided via `Proxy-Authorization`.

```
Header: "SAP-Connectivity-Authentication" : "Bearer <userToken>"
Header: "Proxy-Authorization" : "Bearer <accessToken>"
```



The Cloud Connector validates the token, extracts the available user data, and enables further processing through a configured subject pattern for the resulting short-lived X.509 client certificate.

Back to [Tasks \[page 225\]](#)

1.1.4.1.1.2 Configure Principal Propagation via User Exchange Token

Configure a user exchange token for principal propagation (user propagation) from your Cloud Foundry application to an on-premise system.

Tasks

Task Type	Task
 Operator and/or Developer	Scenario [page 228]
 Developer	Solutions [page 228] Generate the Authentication Token [page 229]

Scenario



For a Cloud Foundry application that uses the Connectivity service, you want the currently logged-in user to be propagated to an on-premise system. For more information, see [Principal Propagation \[page 168\]](#).

Note

When performing a token exchange, the resulting token may lose some of its fields.

Back to [Tasks \[page 227\]](#)

Solutions



You have two options to implement user propagation:

1. **Recommended:** The application sends one header containing the user exchange token to the Connectivity proxy:

```
Header: "Proxy-Authorization" : "Bearer <userExchangeAccessToken>"
```

This option is described in detail in [Generate the Authentication Token \[page 229\]](#) below.

2. The application sends two headers to the Connectivity proxy:

```
Header: "SAP-Connectivity-Authentication" : "Bearer <userToken>"
Header: "Proxy-Authorization" : "Bearer <accessToken>"
```

For more information about this solution, see also [Consuming the Connectivity Service \[page 214\]](#).

Note

This solution is supported to guarantee backward compatibility.

Back to [Tasks \[page 227\]](#)

Generate the Authentication Token



To propagate a user to an on-premise system, you must call the Connectivity proxy using a special JWT (JSON Web token). This token is obtained by exchanging a valid user token following the [OAuth2 JWT Bearer grant type](#).

Note

When using this type of exchange, some of the attributes inside the original user token may not be present in the created JWT.

- [Example: Obtaining a User Token Following the JWT Bearer Grant Type \[page 229\]](#)
- [Example: Calling the Connectivity Proxy with the Exchanged User Token \[page 231\]](#)

Example: Obtaining a User Token Following the JWT Bearer Grant Type

Request:

Sample Code

```
POST https://<host: the value of 'url' from Connectivity service credentials
in VCAP_SERVICES>/oauth/token
Accept: application/json
Content-Type: application/x-www-form-urlencoded
client_id=<connectivity_service_client_id>
client_secret=<connectivity_service_client_secret>
grant_type=urn:iETF:params:oauth:grant-type:jwt-bearer
token_format=jwt
response_type=token
assertion=<logged-in-user-JWT>
```

Response:

Sample Code

```
{
```

```

    "access_token" : "<new-user-JWT>",
    "token_type" : "bearer",
    "expires_in" : 43199,
    "scope" : "<token-scopes>",
    "jti" : "7cc917b8bf6347a2aa18d7ac8f38a1c2"
  }

```

The JWT in `access_token`, also referred to as *user exchange token*, now contains the user details. It is used to consume the Connectivity service.

In case of a Java application, you can use a library that implements the user exchange OAuth flow. Here is an example of how the `userExchangeAccessToken` can be obtained using the [XSUAA Token Client and Token Flow API](#) ↗ :

⚠ Caution

Make sure you get the latest API version.

A sample Maven dependency declaration:

```

<dependency>
  <groupId>com.sap.cloud.security.xsuaa</groupId>
  <artifactId>token-client</artifactId>
  <version><latest version (e.g.: 2.7.7)></version>
</dependency>

```

→ Remember

The XSUAA Token Client library works with multiple HTTP client libraries. Make sure you have one as Maven dependency.

The following sample uses the Apache REST client:

↗ Sample Code

```

// service instance specific OAuth client credentials shall be used
String connectivityServiceClientId = credentials.getString(CLIENT_ID);
String connectivityServiceClientSecret = credentials.getString(CLIENT_SECRET);
// get the URL to xsuaa from the environment variables
URI xsuaaUri = new URI(xsuaaCredentials.getString("token_service_url"));

// use the XSUAA client library to ease the implementation of the user token
exchange flow
XsuaaTokenFlows tokenFlows = new XsuaaTokenFlows(new
DefaultOAuth2TokenService(), new XsuaaDefaultEndpoints(xsUaaUri.toString()),
new ClientCredentials(connectivityServiceClientId,
connectivityServiceClientSecret));
String userExchangeAccessToken =
tokenFlows.userTokenFlow().token(<jwtToken_to_exchange>).execute().getAccessTo
ken();

```

For more information about caching, see also [XSUAA Token Client and Token Flow API - Cache](#) ↗ .

Note

`xsuaaCredentials.getString("token_service_url")` replaces the deprecated property `xsuaaCredentials.getString("url")`, which will be removed soon.

See also: [XSUAA Token Client and Token Flow API](#) .

After obtaining the `userExchangeAccessToken`, you can use it to consume the Connectivity service.

Back to [Generate the Authentication Token \[page 229\]](#)

Example: Calling the Connectivity Proxy with the Exchanged User Token

As a prerequisite to this step, you must configure the Connectivity proxy to be used by your client, see [Set up the HTTP Proxy for On-Premise Connectivity \[page 219\]](#).

Once the application has retrieved the user exchange token, it must pass the token to the Connectivity proxy via the `Proxy-Authorization` header. In this example, we use `URLConnection` as a client.

Sample Code

```
// set user exchange token as Proxy-Authorization header in the URL connection
URLConnection.setRequestProperty("Proxy-Authorization", "Bearer " +
userExchangeAccessToken);
```

Note

Alternatively, you can use the *basic authentication* scheme:

Sample Code

```
// Basic authentication to Connectivity Proxy
String credentials = MessageFormat.format("{0}:{1}", userExchangeAccessToken,
"");
byte[] encodedBytes = Base64.encodeBase64(credentials.getBytes());
URLConnection.setRequestProperty("Proxy-Authorization", "Basic " + new
String(encodedBytes));
```

Back to [Generate the Authentication Token \[page 229\]](#)

Back to [Tasks \[page 227\]](#)

1.1.4.1.1.3 Configure Principal Propagation via IAS Token

Configure an Identity Authentication service (IAS) token for principal propagation (user propagation) from your Cloud Foundry application to an on-premise system.

Scenario

For a Cloud Foundry application that uses the Connectivity service, you want the currently logged-in user to be propagated via the Cloud Connector to an on-premise system. This user is represented in the cloud application by an IAS token.

For more information, see [Principal Propagation \[page 168\]](#) and [Getting Started with the Identity Service of SAP BTP](#).

Prerequisites

- Cloud Connector 2.13 (or newer) must be used.
- The Cloud Connector must be connected to a subaccount that is configured with the IAS tenant issuing the tokens.
For more information, see [Establish Trust and Federation Between SAP Authorization and Trust Management Service and Identity Authentication](#).

Solution

The application sends two headers to the Connectivity proxy:

```
Header: "SAP-Connectivity-Authentication" : "Bearer <IAS-Token>"
```

```
Header: "Proxy-Authorization" : "Bearer <accessToken>"
```



More Information

[Identity Authentication](#)

1.1.4.1.1.4 Configure Principal Propagation via OIDC Token

Configure an OpenID-Connect (OIDC) token for principal propagation (user propagation) from your Cloud Foundry application to an on-premise system.

Tasks

Task Type	Task
 Operator and/or Developer	Scenario [page 233]
 Developer	Solution [page 233]

Scenario



For a Cloud Foundry application that uses the Connectivity service, you want the currently logged-in user to be propagated via the Cloud Connector to an on-premise system. This user is represented in the cloud application by an OIDC token. For more information, see [Principal Propagation \[page 168\]](#) and the [OIDC specification](#) .

Back to [Tasks \[page 233\]](#)

Solution



The Cloud Connector supports principal propagation for OIDC tokens. If on the cloud application side the user is represented by an OIDC token, the application can send the user principal to the Connectivity service (thus reaching the Cloud Connector), using the `SAP-Connectivity-Authentication` HTTP header.

The Cloud Connector validates the token, extracts the available user data, and enables further processing through a configured subject pattern for the resulting short-lived X.509 client certificate.

By default, the user principal is identified by one of the following JWT (JSON web token) attributes:

- `user_name`
- `email`

- mail
- user_uuid
- sub

This list specifies the priority (in descending order from top to bottom) for the default value of `${name}` in the subject pattern of the X.509 client certificate. If a token has more than one of the above claims, the value of `${name}` is extracted from the claim with the highest priority by default.

For the example token below, the default value of `${name}` is `test@user.com`:

Sample Code

```
{
  "aud": "111111111111-2222-3333-444444444444",
  "iss": "https://issuer.com",
  "exp": 2091269073,
  "iat": 1601901108,
  "jti": "111222333444555666777888999000",
  "sub": "test",
  "email": "test@users.com"
}
```

The Cloud Connector administrator can control the exact value to be used as user principal for the *subject CN* of the X.509 client certificate by configuring a subject pattern. For more information, see [Configure Subject Patterns for Principal Propagation \[page 443\]](#).

Back to [Tasks \[page 233\]](#)

1.1.4.1.2 Using the TCP Protocol for Cloud Applications

Access on-premise systems from a Cloud Foundry application via TCP-based protocols, using a SOCKS5 Proxy.

Content

[Concept \[page 234\]](#)

[Restrictions \[page 236\]](#)

[Example \[page 236\]](#)

[Troubleshooting \[page 242\]](#)

Concept

SAP BTP Connectivity provides a SOCKS5 proxy that you can use to access on-premise systems via TCP-based protocols. SOCKS5 is the industry standard for proxying TCP-based traffic (for more information, see IETF RFC 1928).

The SOCKS5 proxy host and port are accessible through the environment variables, which are generated after binding an application to a Connectivity service instance. For more information, see [Consuming the Connectivity Service \[page 214\]](#).

You can access the host under `onpremise_proxy_host`, and the port through `onpremise_socks5_proxy_port`, obtained from the Connectivity service instance.

Authentication to the SOCKS5 proxy is mandatory. It involves the usage of a JWT (JSON Web token) access token (for more information, see IETF RFC 7519). The JWT can be retrieved through the `client_id` and `client_secret`, obtained from the Connectivity service instance. For more information, see [Set up the HTTP Proxy for On-Premise Connectivity \[page 219\]](#), section **Authorization**.

The value of the SOCKS5 protocol authentication method is defined as **0x80** (defined as `X'80'` in IETF, refer to the official specification [SOCKS Protocol Version 5](#)). This value should be sent as part of the authentication method's negotiation request (known as *Initial Request* in SOCKS5). The server then confirms with a response containing its decimal representation (either **128** or **-128**, depending on the client implementation).

After a successful SOCKS5 *Initial Request*, the authentication procedure follows the standard SOCKS5 authentication sub-procedure, that is SOCKS5 *Authentication Request*. The request bytes, in sequence, should look like this:

Bytes	Description
1 byte	Authentication method version - currently 1
4 bytes	Length of the JWT
X bytes	X - The actual value of the JWT in its encoded form
1 byte	Length of the Cloud Connector location ID (0 if no Cloud Connector location ID is used)
Y bytes	Optional. Y - The value of the Cloud Connector location ID in base64-encoded form (if the the value of the location ID is not 0)

The Cloud Connector location ID identifies Cloud Connector instances that are deployed in various locations of a customer's premises and connected to the same subaccount. Since the location ID is an optional property, you should include it in the request only if it has already been configured in the Cloud Connector. For more information, see [Set up Connection Parameters and HTTPS Proxy \[page 391\]](#) (Step 4).

If not set in the Cloud Connector, the byte representing the length of the location ID in the *Authentication Request* should have the value **0**, without including any value for the Cloud Connector location ID (`sccLocationId`).

Back to [Content \[page 234\]](#)

Restrictions

- You cannot use the provided SOCKS5 proxy as general-purpose proxy.
- Proxying UDP traffic is not supported.

Back to [Content \[page 234\]](#)

Example

The following code snippet demonstrates an example based on the Apache Http Client library and Java code, which represents a way to replace the standard socket used in the Apache HTTP client with one that is responsible for authenticating with the Connectivity SOCKS5 proxy:

Sample Code

```
@Override
public void connect(SocketAddress endpoint, int timeout) throws IOException {
    super.connect(getProxyAddress(), timeout);

    OutputStream outputStream = getOutputStream();

    executeSOCKS5InitialRequest(outputStream); // 1. Negotiate authentication
    method, i.e. 0x80 (128)

    executeSOCKS5AuthenticationRequest(outputStream); // 2. Negotiate
    authentication sub-version and send the JWT (and optionally the Cloud
    Connector Location ID)

    executeSOCKS5ConnectRequest(outputStream, (InetSocketAddress)
    endpoint); // 3. Initiate connection to target on-premise backend system
}
```

Sample Code

```
private byte[] createInitialSOCKS5Request() throws IOException {
    ByteArrayOutputStream byteArraysStream = new ByteArrayOutputStream();
    try {
        byteArraysStream.write(SOCKS5_VERSION);
        byteArraysStream.write(SOCKS5_AUTHENTICATION_METHODS_COUNT);
        byteArraysStream.write(SOCKS5_JWT_AUTHENTICATION_METHOD);
        return byteArraysStream.toByteArray();
    } finally {
        byteArraysStream.close();
    }
}

private void executeSOCKS5InitialRequest(OutputStream outputStream) throws
IOException {
    byte[] initialRequest = createInitialSOCKS5Request();
    outputStream.write(initialRequest);

    assertServerInitialResponse();
}
```

Sample Code

```
private byte[] createJWTAuthenticationRequest() throws IOException {
    ByteArrayOutputStream byteArraysStream = new ByteArrayOutputStream();
    try {
        byteArraysStream.write(SOCKS5_JWT_AUTHENTICATION_METHOD_VERSION);

        byteArraysStream.write(ByteBuffer.allocate(4).putInt( jwtToken.getBytes().length
        h).array());
        byteArraysStream.write( jwtToken.getBytes());
        byteArraysStream.write(ByteBuffer.allocate(1).put((byte)
        sccLocationId.getBytes().length).array());
        byteArraysStream.write(sccLocationId.getBytes());
        return byteArraysStream.toByteArray();
    } finally {
        byteArraysStream.close();
    }
}

private void executeSOCKS5AuthenticationRequest(OutputStream outputStream)
throws IOException {
    byte[] authenticationRequest = createJWTAuthenticationRequest();
    outputStream.write(authenticationRequest);

    assertAuthenticationResponse();
}
}
```

In version 4.2.6 of the Apache HTTP client, the class responsible for connecting the socket is `DefaultClientConnectionOperator`. By extending the class and replacing the standard socket with the complete example code below, which implements a Java Socket, you can handle the SOCKS5 authentication with ID **0x80**. It is based on a JWT and supports the Cloud Connector location ID.

Sample Code

```
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.SocketAddress;
import java.net.SocketException;
import java.nio.ByteBuffer;

import java.util.Base64; // or any other library for base64 encoding
import org.json.JSONArray; // or any other library for JSON objects
import org.json.JSONObject; // or any other library for JSON objects
import org.json.JSONException; // or any other library for JSON objects

public class ConnectivitySocks5ProxySocket extends Socket {

    private static final byte SOCKS5_VERSION = 0x05;
    private static final byte SOCKS5_JWT_AUTHENTICATION_METHOD = (byte) 0x80;
    private static final byte SOCKS5_JWT_AUTHENTICATION_METHOD_VERSION = 0x01;
    private static final byte SOCKS5_COMMAND_CONNECT_BYTE = 0x01;
    private static final byte SOCKS5_COMMAND_REQUEST_RESERVED_BYTE = 0x00;
    private static final byte SOCKS5_COMMAND_ADDRESS_TYPE_IPv4_BYTE = 0x01;
    private static final byte SOCKS5_COMMAND_ADDRESS_TYPE_DOMAIN_BYTE = 0x03;
    private static final byte SOCKS5_AUTHENTICATION_METHODS_COUNT = 0x01;
    private static final int SOCKS5_JWT_AUTHENTICATION_METHOD_UNSIGNED_VALUE
    = 0x80 & 0xFF;
}
```

```

private static final byte SOCKS5_AUTHENTICATION_SUCCESS_BYTE = 0x00;

private static final String SOCKS5_PROXY_HOST_PROPERTY =
"onpremise_proxy_host";
private static final String SOCKS5_PROXY_PORT_PROPERTY =
"onpremise_socks5_proxy_port";

private final String jwtToken;
private final String sccLocationId;

public ConnectivitySocks5ProxySocket(String jwtToken, String
sccLocationId) {
    this.jwtToken = jwtToken;
    this.sccLocationId = sccLocationId != null ?
Base64.getEncoder().encodeToString(sccLocationId.getBytes()) : "";
}

protected InetSocketAddress getProxyAddress() {
    try {
        JSONObject credentials = extractEnvironmentCredentials();
        String proxyHost =
credentials.getString(SOCKS5_PROXY_HOST_PROPERTY);
        int proxyPort =
Integer.parseInt(credentials.getString(SOCKS5_PROXY_PORT_PROPERTY));
        return new InetSocketAddress(proxyHost, proxyPort);
    } catch (JSONException ex) {
        throw new IllegalStateException("Unable to extract the SOCKS5
proxy host and port", ex);
    }
}

private JSONObject extractEnvironmentCredentials() throws JSONException {
    JSONObject jsonObj = new JSONObject(System.getenv("VCAP_SERVICES"));
    JSONArray jsonArr = jsonObj.getJSONArray("connectivity");
    return jsonArr.getJSONObject(0).getJSONObject("credentials");
}

@Override
public void connect(SocketAddress endpoint, int timeout) throws
IOException {
    super.connect(getProxyAddress(), timeout);

    OutputStream outputStream = getOutputStream();

    executeSOCKS5InitialRequest(outputStream);

    executeSOCKS5AuthenticationRequest(outputStream);

    executeSOCKS5ConnectRequest(outputStream, (InetSocketAddress)
endpoint);
}

private void executeSOCKS5InitialRequest(OutputStream outputStream)
throws IOException {
    byte[] initialRequest = createInitialSOCKS5Request();
    outputStream.write(initialRequest);

    assertServerInitialResponse();
}

private byte[] createInitialSOCKS5Request() throws IOException {
    ByteArrayOutputStream byteArraysStream = new ByteArrayOutputStream();
    try {
        byteArraysStream.write(SOCKS5_VERSION);
        byteArraysStream.write(SOCKS5_AUTHENTICATION_METHODS_COUNT);
        byteArraysStream.write(SOCKS5_JWT_AUTHENTICATION_METHOD);
        return byteArraysStream.toByteArray();
    } finally {

```

```

        byteArraysStream.close();
    }
}

private void assertServerInitialResponse() throws IOException {
    InputStream inputStream = getInputStream();

    int versionByte = inputStream.read();
    if (SOCKS5_VERSION != versionByte) {
        throw new SocketException(String.format("Unsupported SOCKS
version - expected %s, but received %s", SOCKS5_VERSION, versionByte));
    }

    int authenticationMethodValue = inputStream.read();
    if (SOCKS5_JWT_AUTHENTICATION_METHOD_UNSIGNED_VALUE !=
authenticationMethodValue) {
        throw new SocketException(String.format("Unsupported
authentication method value - expected %s, but received %s",
SOCKS5_JWT_AUTHENTICATION_METHOD_UNSIGNED_VALUE,
authenticationMethodValue));
    }
}

private void executeSOCKS5AuthenticationRequest(OutputStream
outputStream) throws IOException {
    byte[] authenticationRequest = createJWTAuthenticationRequest();
    outputStream.write(authenticationRequest);

    assertAuthenticationResponse();
}

private byte[] createJWTAuthenticationRequest() throws IOException {
    ByteArrayOutputStream byteArraysStream = new ByteArrayOutputStream();
    try {
        byteArraysStream.write(SOCKS5_JWT_AUTHENTICATION_METHOD_VERSION);

byteArraysStream.write(ByteBuffer.allocate(4).putInt(jwtToken.getBytes().length).array());
        byteArraysStream.write(jwtToken.getBytes());
        byteArraysStream.write(ByteBuffer.allocate(1).put((byte)
sccLocationId.getBytes().length).array());
        byteArraysStream.write(sccLocationId.getBytes());
        return byteArraysStream.toByteArray();
    } finally {
        byteArraysStream.close();
    }
}

private void assertAuthenticationResponse() throws IOException {
    InputStream inputStream = getInputStream();

    int authenticationMethodVersion = inputStream.read();
    if (SOCKS5_JWT_AUTHENTICATION_METHOD_VERSION !=
authenticationMethodVersion) {
        throw new SocketException(String.format("Unsupported
authentication method version - expected %s, but received %s",
SOCKS5_JWT_AUTHENTICATION_METHOD_VERSION,
authenticationMethodVersion));
    }

    int authenticationStatus = inputStream.read();
    if (SOCKS5_AUTHENTICATION_SUCCESS_BYTE != authenticationStatus) {
        throw new SocketException("Authentication failed!");
    }
}

private void executeSOCKS5ConnectRequest(OutputStream outputStream,
InetSocketAddress endpoint) throws IOException {

```

```

byte[] commandRequest = createConnectCommandRequest(endpoint);
outputStream.write(commandRequest);

assertConnectCommandResponse();
}

private byte[] createConnectCommandRequest(InetSocketAddress endpoint)
throws IOException {
    String host = endpoint.getHostName();
    int port = endpoint.getPort();
    ByteArrayOutputStream byteArraysStream = new ByteArrayOutputStream();
    try {
        byteArraysStream.write(SOCKS5_VERSION);
        byteArraysStream.write(SOCKS5_COMMAND_CONNECT_BYTE);
        byteArraysStream.write(SOCKS5_COMMAND_REQUEST_RESERVED_BYTE);
        byte[] hostToIPv4 = parseHostToIPv4(host);
        if (hostToIPv4 != null) {
            byteArraysStream.write(SOCKS5_COMMAND_ADDRESS_TYPE_IPv4_BYTE);
            byteArraysStream.write(hostToIPv4);
        } else {
            byteArraysStream.write(SOCKS5_COMMAND_ADDRESS_TYPE_DOMAIN_BYTE);
            byteArraysStream.write(ByteBuffer.allocate(1).put((byte)
host.getBytes().length).array());
            byteArraysStream.write(host.getBytes());
        }
        byteArraysStream.write(ByteBuffer.allocate(2).putShort((short)
port).array());
        return byteArraysStream.toByteArray();
    } finally {
        byteArraysStream.close();
    }
}

private void assertConnectCommandResponse() throws IOException {
    InputStream inputStream = getInputStream();

    int versionByte = inputStream.read();
    if (SOCKS5_VERSION != versionByte) {
        throw new SocketException(String.format("Unsupported SOCKS
version - expected %s, but received %s", SOCKS5_VERSION, versionByte));
    }

    int connectStatusByte = inputStream.read();
    assertConnectStatus(connectStatusByte);

    readRemainingCommandResponseBytes(inputStream);
}

private void assertConnectStatus(int commandConnectStatus) throws
IOException {
    if (commandConnectStatus == 0) {
        return;
    }

    String commandConnectStatusTranslation;
    switch (commandConnectStatus) {
        case 1:
            commandConnectStatusTranslation = "FAILURE";
            break;
        case 2:
            commandConnectStatusTranslation = "FORBIDDEN";
            break;
        case 3:
            commandConnectStatusTranslation = "NETWORK_UNREACHABLE";
            break;
        case 4:
            commandConnectStatusTranslation = "HOST_UNREACHABLE";

```

```

        break;
    case 5:
        commandConnectStatusTranslation = "CONNECTION_REFUSED";
        break;
    case 6:
        commandConnectStatusTranslation = "TTL_EXPIRED";
        break;
    case 7:
        commandConnectStatusTranslation = "COMMAND_UNSUPPORTED";
        break;
    case 8:
        commandConnectStatusTranslation = "ADDRESS_UNSUPPORTED";
        break;
    default:
        commandConnectStatusTranslation = "UNKNOWN";
        break;
    }
    throw new SocketException("SOCKS5 command failed with status: " +
commandConnectStatusTranslation);
}

private byte[] parseHostToIPv4(String hostName) {
    byte[] parsedHostName = null;
    String[] virtualHostOctets = hostName.split("\\.", -1);
    int octetsCount = virtualHostOctets.length;
    if (octetsCount == 4) {
        try {
            byte[] ipOctets = new byte[octetsCount];
            for (int i = 0; i < octetsCount; i++) {
                int currentOctet = Integer.parseInt(virtualHostOctets[i]);
                if ((currentOctet < 0) || (currentOctet > 255)) {
                    throw new
IllegalArgumentException(String.format("Provided octet %s is not in the range
of [0-255]", currentOctet));
                }
                ipOctets[i] = (byte) currentOctet;
            }
            parsedHostName = ipOctets;
        } catch (IllegalArgumentException ex) {
            return null;
        }
    }

    return parsedHostName;
}


private void readRemainingCommandResponseBytes(InputStream inputStream)
throws IOException {
    inputStream.read(); // skipping over SOCKS5 reserved byte
    int addressTypeByte = inputStream.read();
    if (SOCKS5_COMMAND_ADDRESS_TYPE_IPv4_BYTE == addressTypeByte) {
        for (int i = 0; i < 6; i++) {
            inputStream.read();
        }
    } else if (SOCKS5_COMMAND_ADDRESS_TYPE_DOMAIN_BYTE ==
addressTypeByte) {
        int domainNameLength = inputStream.read();
        int portBytes = 2;
        inputStream.read(new byte[domainNameLength + portBytes], 0,
domainNameLength + portBytes);
    }
}
}

```

[Back to Example \[page 236\]](#)

[Back to Content \[page 234\]](#)

Troubleshooting

If the handshake with the SOCKS5 proxy server fails, a SOCKS5 protocol error is returned, see IETF [RFC 1928](#) . The table below shows the most common errors and their root cause in the scenario you use:




SOCKS5 Error Code	Technical Description	Client-Side Error Description	Scenario Error
0x00	SUCCESS		Success
0x01	FAILURE		Connection closed by backend or general scenario failure.
0x02	FORBIDDEN	Connection not allowed by ruleset	No matching host mapping found in Cloud Connector access control settings, see Configure Access Control (TCP) [page 474] .
0x03	NETWORK_UNREACHABLE		The Cloud Connector is not connected to the subaccount and the Cloud Connector Location ID that is used by the cloud application can't be identified. See Connect and Disconnect a Cloud Subaccount [page 658] and Managing Subaccounts [page 401] , section <i>Procedure</i> .
0x04	HOST_UNREACHABLE		Cannot open connection to the backend, that is, the host is unreachable.
0x05	CONNECTION_REFUSED		Authentication failure
0x06	TTL_EXPIRED		<i>Not used</i>
0x07	COMMAND_UNSUPPORTED		Only the SOCKS5 CONNECT command is supported.
0x08	ADDRESS_UNSUPPORTED		Only the SOCKS5 DOMAIN and IPv4 commands are supported.

[Back to Content \[page 234\]](#)

1.1.4.2 Consuming the Destination Service

Retrieve and store externalized technical information about the destination to consume a target remote service from your Cloud Foundry application.

Tasks

Task Type	Task
 Operator and/or Developer	Overview [page 243] <hr/> Prerequisites [page 244]
 Developer	Steps [page 245] <ol style="list-style-type: none">1. Read Credentials from the Environment Variables [page 245]2. Generate a JSON Web Token (JWT) [page 246]3. Call the Destination Service [page 247]
 Operator and/or Developer	Destination Configuration Attributes [page 250]

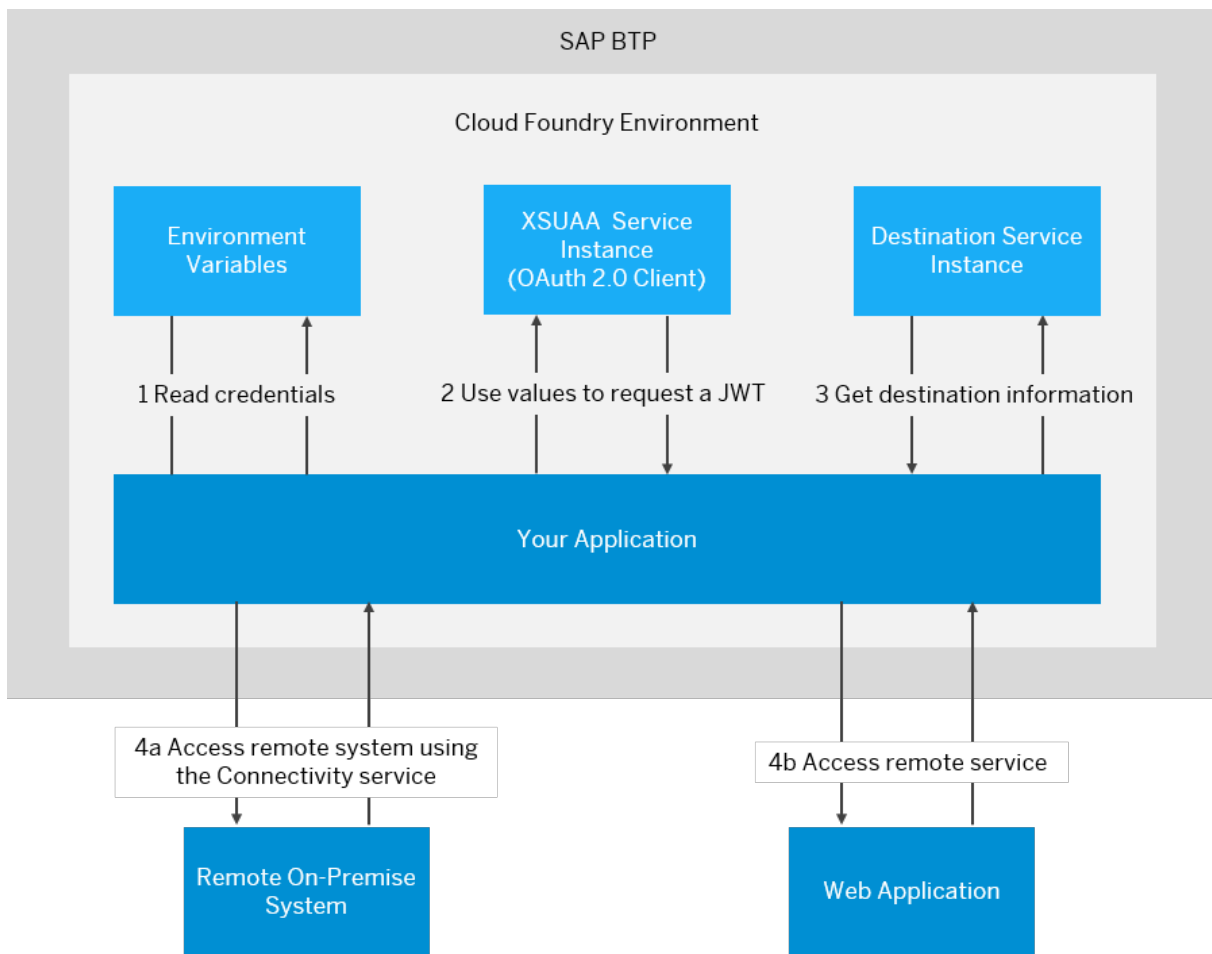
Overview



The Destination service lets you find the destination information that is required to access a remote service or system from your Cloud Foundry application.

- For the connection to an *on-premise system*, you can optionally use this service, together with (i.e. in addition to) the Connectivity service, see [Consuming the Connectivity Service \[page 214\]](#).
- For the connection to any other *Web application* (remote service), you can use the Destination service without the Connectivity service.

Consuming the Destination Service includes user authorization via a JSON Web Token (JWT) that is provided by the *xsuaa* service.



Back to [Tasks \[page 243\]](#)

Prerequisites



- To manage destinations and certificates on service instance level (all CRUD operations), you must be assigned to one of the following roles: `OrgManager`, `SpaceManager` or `SpaceDeveloper`.

Note

The role `SpaceAuditor` has only `Read` permission for destinations and certificates.

- To consume the Destination service from an application, you must create a service instance and bind it to the application. See [Create and Bind a Destination Service Instance \[page 209\]](#).
- To generate the required JSON Web Token (JWT), you must bind the application to an instance of the `xsuaa` service using the service plan 'application'. The `xsuaa` service instance acts as an OAuth 2.0 client

and grants user access to the bound application. Make sure that you set the `xsappid` property when creating the instance. Find a detailed guide for this procedure in section 3. *Creation of the Authorization & Trust Management Instance (aka XSUAA)* of the SCN blog [How to use SAP BTP Connectivity and Cloud Connector in the Cloud Foundry environment](#).

- You need at least one configured destination, otherwise there will be nothing to retrieve via the service. To access the *Destinations* editor in the cockpit, follow the steps in [Access the Destinations Editor \[page 61\]](#). To manage destinations via REST API, see [Destination Service REST API \[page 80\]](#).

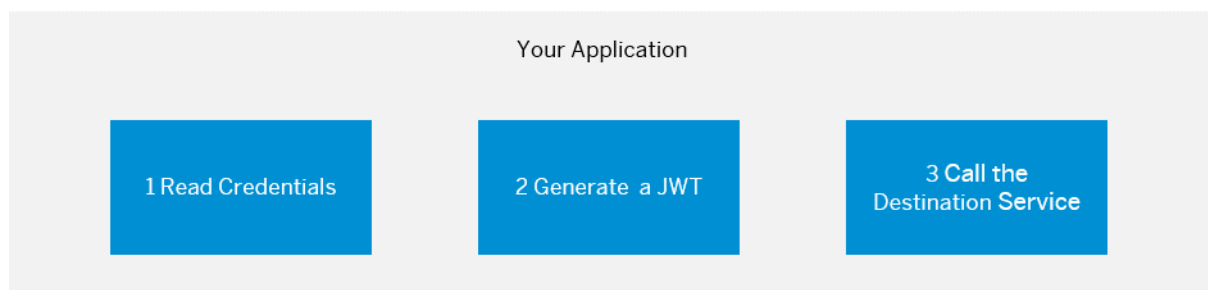
Back to [Tasks \[page 243\]](#)

Steps



To consume the Destination service from your application, perform the following basic steps:

1. [Read Credentials from the Environment Variables \[page 245\]](#)
2. [Generate a JSON Web Token \(JWT\) \[page 246\]](#)
3. [Call the Destination Service \[page 247\]](#)



Back to [Tasks \[page 243\]](#)

Read Credentials from the Environment Variables



The Destination service stores its credentials in the environment variables. To consume the service, you require the following information:

- The value of `clientid`, `clientsecret` and `uri` from the Destination service credentials.

- The values of `url` from the `xsuaa` credentials.

You can access this information as follows:

- From the CLI, the following command lists the environment variables of `<app-name>`:

```
cf env <app-name>
```

- From within the application, the service credential can be accessed as described in [Consuming the Connectivity Service \[page 214\]](#).

Note

Below, we refer to the `JSONObjects`, containing the instance credentials as `destinationCredentials` (for the Destination service) and `xsuaaCredentials` (for `xsuaa`).

Back to [Tasks \[page 243\]](#)

Generate a JSON Web Token (JWT)



Your application must create an OAuth client using the attributes `clientId` and `clientSecret`, which are provided by the Destination service instance. Then, you must retrieve a new JWT from UAA and pass it in the `Authorization` HTTP header.

Two examples how to achieve this (Java and cURL):

Java:

For a Java application, you can use a library that implements the client credentials OAuth flow. Here is an example of how the `clientCredentialsTokenFlow` can be obtained using the [XSUAA Token Client and Token Flow API](#) 🐘:

Caution

Make sure you get the latest API version.

A sample Maven dependency declaration:

```
<dependency>
  <groupId>com.sap.cloud.security.xsuaa</groupId>
  <artifactId>token-client</artifactId>
  <version><latest version (e.g.: 2.7.7)></version>
</dependency>
```

→ Remember

The XSUAA Token Client library works with multiple HTTP client libraries. Make sure you have one as Maven dependency.

The following sample uses the Apache REST client:

↔ Sample Code

```
// get value of "clientid" and "clientsecret" from the environment variables
String clientid = destinationCredentials.getString("clientid");
String clientsecret = destinationCredentials.getString("clientsecret");

// get the URL to xsuaa from the environment variables
URI xsuaaUri = new URI(xsuaaCredentials.getString("url"));

// use the XSUAA client library to ease the implementation of the user token
exchange flow
XsuaaTokenFlows tokenFlows = new XsuaaTokenFlows(new
DefaultOAuth2TokenService(), new XsuaaDefaultEndpoints(xsUaaUri.toString()),
new ClientCredentials(clientid, clientsecret));

String jwtToken =
tokenFlows.clientCredentialsTokenFlow().execute().getAccessToken();
```

For more information about caching, see also [XSUAA Token Client and Token Flow API - Cache](#).

cURL:

↔ Sample Code

```
curl -X POST \
  <xsuaa-url>/oauth/token \
  -H 'authorization: Basic <<clientid>:<clientsecret> encoded with Base64>' \
  -H 'content-type: application/x-www-form-urlencoded' \
  -d 'client_id=<clientid>&grant_type=client_credentials'
```

Back to [Tasks \[page 243\]](#)

Call the Destination Service



When calling the Destination service, use the `uri` attribute, provided in `VCAP_SERVICES`, to build the request URLs.

[Read a Destination by only Specifying its Name \("Find Destination"\) \[page 248\]](#)

[Read a Destination Associated with a Subaccount \[page 248\]](#)

[Get All Destinations Associated with a Subaccount \[page 249\]](#)

[Response Codes \[page 250\]](#)

Read a Destination by only Specifying its Name ("Find Destination")

This lets you provide simply a name of the destination while the service will search for it. First, the service searches the destinations that are associated with the service instance. If none of the destinations match the requested name, the service searches the destinations that are associated with the subaccount.

- Path: `/destination-configuration/v1/destinations/<destination-name>`
- Example of a call (cURL):

Sample Code

```
curl "<uri>/destination-configuration/v1/destinations/<destination-name>" \
-X GET \
-H "Authorization: Bearer <jwtToken>"
```

- Example of a response (this is a destination found when going through the subaccount destinations):

Sample Code

```
{
  "owner":
  {
    "SubaccountId":<id>,
    "InstanceId":null
  },
  "destinationConfiguration":
  {
    "Name": "demo-internet-destination",
    "URL": "http://www.google.com",
    "ProxyType": "Internet",
    "Type": "HTTP",
    "Authentication": "NoAuthentication"
  }
}
```

Note

The response from this type of call contains not only the configuration of the requested destination, but also some additional data. See ["Find Destination" Response Structure \[page 259\]](#).

Back to [Call the Destination Service \[page 247\]](#)

Read a Destination Associated with a Subaccount

This lets you retrieve the configurations of a destination that is defined within a subaccount, by providing the name of the destination.

- Path: `/destination-configuration/v1/subaccountDestinations/<destination-name>`

- Example of a call (cURL):

Sample Code

```
curl "<uri>/destination-configuration/v1/subaccountDestinations/
<destination name>" \
-X GET \
-H "Authorization: Bearer <jwtToken>"
```

- Example of a response:

Sample Code

```
{
  "Name": "demo-internet-destination",
  "URL": "http://www.google.com",
  "ProxyType": "Internet",
  "Type": "HTTP",
  "Authentication": "NoAuthentication"
}
```

Back to [Call the Destination Service \[page 247\]](#)

Get All Destinations Associated with a Subaccount

This lets you retrieve the configurations of all destinations that are defined within a subaccount.

- Path: /destination-configuration/v1/subaccountDestinations
- Example of a call (cURL):

Sample Code

```
curl "<uri>/destination-configuration/v1/subaccountDestinations" \
-X GET \
-H "Authorization: Bearer <jwtToken>"
```

- Example of a response:

Sample Code

```
[
  {
    "Name": "demo-onpremise-destination1",
    "URL": "http://virtualhost:1234",
    "ProxyType": "OnPremise",
    "Type": "HTTP",
    "Authentication": "NoAuthentication"
  },
  {
    "Name": "demo-onpremise-destination2",
    "URL": "http://virtualhost:4321",
    "ProxyType": "OnPremise",
    "Type": "HTTP",
    "Authentication": "BasicAuthentication",
    "User": "myname123",
    "Password": "123456"
  }
]
```


]

Back to [Call the Destination Service \[page 247\]](#)

Response Codes

When calling the Destination service, you may get the following response codes:

- **200:** OK (Json of Destination)
- **401:** Unauthorized (Authentication Failed)
- **403:** Forbidden (Authorization Failed)
- **404:** The requested destination could not be found (not applicable to 'get all destinations associated with a subaccount')
- **500:** Internal Server Error

Back to [Call the Destination Service \[page 247\]](#)

Back to [Tasks \[page 243\]](#)

Destination Configuration Attributes



The JSON object that serves as the response of a successful request (value of the `destinationConfiguration` property for "Find destination") can have different attributes, depending on the authentication type and proxy type of the corresponding destination. See [HTTP Destinations \[page 86\]](#).

Back to [Tasks \[page 243\]](#)

Related Information

[User Propagation via SAML 2.0 Bearer Assertion Flow \[page 251\]](#)

[Destination Service REST API \[page 80\]](#)

[Exchanging User JWTs via OAuth2UserTokenExchange Destinations \[page 263\]](#)

[Use Cases \[page 280\]](#)

[Multitenancy in the Destination Service \[page 265\]](#)



[Destination Java APIs \[page 267\]](#)

[Extending Destinations with Fragments \[page 273\]](#)

1.1.4.2.1 User Propagation via SAML 2.0 Bearer Assertion Flow

Learn about the process for automatic token retrieval, using the `OAuth2SAMLBearerAssertion` authentication type for HTTP destinations.

Tasks

Task Type	Task
 Operator and/or Developer	Prerequisites [page 251]
 Developer	Automated Access Token Retrieval [page 252] <ul style="list-style-type: none">• Determine the Propagated User ID [page 252]• Propagate User Attributes [page 254]• Scenarios [page 254]

Prerequisites



- You have configured an `OAuth2SAMLBearerAssertion` destination. See [OAuth SAML Bearer Assertion Authentication \[page 94\]](#).
- Unless using the destination property `SystemUser`, the user's identity should be represented by a JSON Web token (JWT).

⚠ Caution

The `SystemUser` property is deprecated and will be removed soon. We recommend that you work on behalf of specific (named) users instead of working with a technical user.

As an alternative for technical user communication, we strongly recommend that you use one of these authentication types:

- Basic Authentication (see [Client Authentication Types for HTTP Destinations \[page 101\]](#))
- Client Certificate Authentication (see [Client Authentication Types for HTTP Destinations \[page 101\]](#))
- [OAuth Client Credentials Authentication \[page 104\]](#)

To extend an OAuth access token's validity, consider using an OAuth refresh token.

Note

Though actually not being a strict requirement, it is likely that you need a user JWT to get the relevant information. See [SAP Authorization and Trust Management Service](#).

- If you are using custom user attributes to determine the user, the JWT representing the user (that is passed to the Destination service) must have the `user_attributes` scope.

Back to [Tasks \[page 251\]](#)

Automated Access Token Retrieval



For an `OAuth2SAMLBearerAssertion` destination, you can use the automated token retrieval functionality that is available via the "find destination" endpoint. See [Destination Service REST API \[page 80\]](#).

Determine the Propagated User ID

There are currently three sources that can provide the propagated user ID. They are prioritized, meaning that the lookup always starts from the top-priority source and goes down the list. If the propagated user ID is not found at a given level, the next level is checked. If not found on any level, the operation would fail.

Find the available sources in the table below, in order of their priority.

Propagated User ID: Sources

Source	Procedure
<i>System User</i>	The system user is a special user ID that is hardcoded in your destination as value of the <code>SystemUser</code> property. If you set this property, its value is used as the propagated user ID.

Source

Field in the JWT

Procedure

In this case, the Destination service looks for the user ID as a field in the provided JWT. When you make the HTTP call to the Destination service, you must provide the `Authorization` header. The value must be a JWT in its encoded form (see [RFC 7519](#)). The procedure is as follows:

- If the `userIdSource` property is configured in the destination, its value is the key of the JWT field that will be the user ID (if there is no such key in the JWT, the flow proceeds to the next level). There are 2 options:
 - plain string: the exact match is searched on the root-level element keys of the JWT.
 - [JsonPath](#) expression: lets you use non-root-level elements of the JWT.
- If the `userIdSource` property is missing, the flow falls back to the destination property `nameIdFormat`. It must have one of the following two values (or not be set at all), otherwise an exception is thrown:
 - **urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress**: the `email` element of the JWT is the user ID. If there is no such element in the JWT, an exception is thrown.
 - **urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified** (or not set): the `user_name` element of the JWT is the user ID. If there is no such element in the JWT, an exception is thrown.

Custom User Attribute

Like *Field in the JWT*, this source must use the `Authorization` header. In this case, its value is used to retrieve the custom user attributes from the Identity Provider (XSUAA). One of those attributes can be used as the propagated user ID. The access token from the header must be a user JWT with the `user_attributes` scope. Otherwise the custom attributes cannot be retrieved, and the operation results in an error.

- If the `userIdSource` property is configured in the destination, the same logic applies as for *Field in the JWT*, but this time on the JSON containing the custom user attributes.
- If `userIdSource` is missing or the desired key is not found in the custom attributes, the operation fails (`user ID could not be determined`).

[Back to Tasks \[page 251\]](#)

Propagate User Attributes

You can read additional user attributes from the identity provider (XSUAA), and propagate them as SAML attributes in the generated SAML bearer assertion.

These attributes are similar to the ones returned by the Cloud Foundry UAA user info endpoint. However, they may differ depending on the XSUAA behavior, and are specific to the identity provider you use.

For more details about these attributes and possible values, see <https://docs.cloudfoundry.org/api/uaa/version/74.15.0/index.html#user-info>.

Note

When adding the attributes, the following rules apply:

- All root elements, except for `user_attributes`, are added "as is", that is, the attribute name and value are displayed as seen in the source (user info response structure).
- Elements under the `user_attribute` key are parsed and added as attributes prefixed with `'user_attributes.'`. For example, having `{ "user_attributes": { "my_param": "my_value" } }` will result in an attribute called `'user_attributes.my_param'` with value `'my_value'` in the SAML assertion. If you want to avoid this `user_attributes.` prefix, you can set the `skipUserAttributesPrefixInSAMLAttributes` additional property of the destination to `true`. If you do so, the above example will result in an attribute called `my_param` with value `my_value` in the SAML assertion.

In addition to identity provider (XSUAA) user info attributes, there are some more attributes which are read from the passed JWT. They are located via predefined `JsonPath` expressions and cannot be controlled by the end user:

- `$.['xs.system.attributes']['xs.saml.groups']`
- `$.['user_attributes']['xs.saml.groups']`

Note

The `'xs.saml.groups'` attribute, read from the passed JWT, is renamed to `'Groups'` in the resulting SAML assertion. See also [Settings for Default SAML Federation Attributes of Identity Providers for Business Users](#).

Back to [Tasks \[page 251\]](#)

Scenarios

Refer to the table below to find the JWT requirements for a specific scenario:

Scenario	Authorization Header
Propagate a technical user principal, using the <code>SystemUser</code> property of an <code>OAuth2SAMLBearerAssertion</code> destination maintained in the subscriber subaccount , and used by the provider application .	Access token, retrieved <ul style="list-style-type: none"> via the client credentials of the Destination service instance (bound to the application). using the subscriber's tenant-specific <code>Token Service URL</code>.
Propagate a technical user principal, using the <code>SystemUser</code> property of an <code>OAuth2SAMLBearerAssertion</code> destination maintained in the same subaccount where the application is deployed.	Access token, retrieved <ul style="list-style-type: none"> via the client credentials of the Destination service instance (bound to the application). using the provider's tenant-specific <code>Token Service URL</code>.
Propagate a business user principal, using an <code>OAuth2SAMLBearerAssertion</code> destination maintained in the subscriber subaccount where the application is deployed. The business user is represented by a JWT that was issued by the subscriber .	The JWT, previously retrieved from the application <ul style="list-style-type: none"> by exchanging the JWT (that represents the user) to another user access token via the client credentials of the Destination service instance (bound to the application). using the subscriber's tenant-specific <code>Token Service URL</code>.
Propagate a business user principal, using an <code>OAuth2SAMLBearerAssertion</code> destination maintained in the same subaccount where the application is deployed. The business user is represented by a JWT that was issued by the provider .	The JWT, previously retrieved by the application <ul style="list-style-type: none"> by exchanging the JWT (that represents the user) to another user access token via the client credentials of the Destination service instance (bound to the application). using the provider's tenant-specific <code>Token Service URL</code>.

Back to [Tasks \[page 251\]](#)

1.1.4.2.2 Destination Service REST API

Destination service REST API specification for the SAP Cloud Foundry environment.

The Destination service provides a REST API that you can use to read and manage resources like destinations, certificates and destination fragments on all available levels. This API is documented in the [SAP Business Accelerator Hub](#).

It shows all available endpoints, the supported operations, parameters, possible error cases and related status codes, etc. You can also execute requests using the credentials (for example, the service key) of your Destination service instance, see [Create and Bind a Destination Service Instance \[page 209\]](#).

Related Information

[Calling the Destination Service REST API \[page 256\]](#)

["Find Destination" Response Structure \[page 259\]](#)

1.1.4.2.2.1 Calling the Destination Service REST API

Prerequisites and steps to get access to the Destination service REST API.

Prerequisites

To call the Destination service REST API, you must have a Destination service instance inside your subaccount.

If you:

- **don't have** such instance, follow the next step to get it up and running.
- **have** such instance, you can skip the next step and go to [Get Credentials \[page 257\]](#).

1. Set up a Destination Service Instance for your Subaccount

To create a Destination service instance inside your subaccount, see [Creating Service Instances](#) about creating service instances in the **BTP cockpit**, or from the **Cloud Foundry CLI**.

Note

When creating a Destination service instance, you can refer to the following yaml segment for the basic information of the instance.

Basic information for the Destination service instance

- *Plan*: lite
Currently, the Destination service offers only this plan.
 - *Runtime Environment*: Cloud Foundry
Cloud Foundry must be enabled for your subaccount.
 - *Space*: <space_name>
Choose the space in which the Destination service instance will reside.
 - *Instance Name*: <instance_name>
Enter a name of your choice here for the instance.
-

Note

The yaml segment above contains Cloud Foundry-specific information. Other configurations are possible, for instance when running on Kyma. The important part is to deploy your instance in the target runtime and get the service key / binding credentials to access the Destination service.

2. Get Credentials

To access the Destination service REST API, you need an **access token**. To generate this token, you need credentials contained in a **service key** of the Destination service instance.

If you don't have any service keys in your Destination service instance, see [Creating Service Keys](#) about creating a service key for an instance in the **BTP cockpit** or from the **Cloud Foundry CLI**.

Once you have a service key for your Destination service instance, you open it and extract the following information:

Information to extract from the service key

- `clientid: "<value_to_extract>"`
The client ID that will be used for authentication in the next step.
 - `clientsecret: "<value_to_extract>"`
The client secret that will be used for authentication in the next step.
 - `url: "<value_to_extract>"`
The authentication endpoint from which you get an access token for the Destination service.
 - `uri: "<value_to_extract>"`
The URL of the Destination service.
-

Note

For demonstration purposes, we are using the `clientid` and `clientsecret` for authentication. We recommend that you use mTLS which uses an X.509 client certificate because it is a more secure way of authenticating.

3. Get an AccessToken

In this step, you will get an **access token** from XSUAA which you can then use to authenticate towards the Destination service REST API. For this step, you must use the values you extracted for **clientid**, `clientsecret` and **url** from the previous step.

Here is an example call using **curl**:

Sample Code

```
curl -X POST \  
  "<url>/oauth/token" \  
  -H "Content-Type: application/x-www-form-urlencoded" \  
  -d "grant_type=client_credentials" --data-urlencode \  
  "client_id=<client_id>" --data-urlencode "client_secret=<client_secret>"
```

where:

- <url> is the value of the URL from the previous step.
- <client_id> is the value of *clientid* from the previous step.
- <client_secret> is the value of *clientsecret* from the previous step.

The access token is shown in the **access_token** key in the response JSON. Make sure you save it because you will need it in the next step.

4. Call the Destination Service REST API

Now that you have an access token for the Destination service, you can call one of the Destination service REST API endpoints. To get the full list of available endpoints in the Destination service REST API, see the [Destination Service REST API reference](#) on SAP Business Accelerator Hub.

Here is an example of the call using **curl**:

Sample Code

```
curl -X GET \  
  "<uri>/destination-configuration/v1/<endpoint>" \  
  -H "Authorization: Bearer <access_token>"
```

where:

- <uri> is the value of *uri* from [Get Credentials \[page 257\]](#).
- <endpoint> is the endpoint of the Destination service REST API that you want to call.
- <access_token> is the access token you saved from the previous section.

If, for example, you want to make a GET call towards the */subaccountDestinations* endpoint, the call would look like:

Sample Code

```
curl -X GET \  
  "<uri>/destination-configuration/v1/subaccountDestinations" \  
  -H "Authorization: Bearer <access_token>"
```

Example: Response from the Destination service

Sample Code

```
[
  {
    "Name": "no-authentication-destination",
    "Type": "HTTP",
    "URL": "https://sap.com",
    "Authentication": "NoAuthentication",
    "ProxyType": "Internet"
  },
  {
    "Name": "basic-authentication-destination",
    "Type": "HTTP",
    "URL": "https://sap.com",
    "Authentication": "BasicAuthentication",
    "ProxyType": "Internet",
    "User": "my-user",
    "Password": "my-password"
  }
]
```

1.1.4.2.2.2 "Find Destination" Response Structure

Overview of data that are returned by the Destination service for the call type "find destination".

Response Structure

When you use the "find destination" call (read a destination by only specifying its name), the structure of the response includes four parts:

- The [owner of the destination \[page 259\]](#).
- The actual [destination configuration \[page 260\]](#).
- (Optional) [Authentication tokens \[page 260\]](#) that are relevant to the destination.
- (Optional) [Certificates \[page 262\]](#) that are relevant to the destination.

Each of these parts is represented in the JSON object as a key-value pair and their values are JSON objects, see [Example \[page 262\]](#).

See also [Call the Destination Service \[page 247\]](#).

Destination Owner

- **Key:** owner

The JSON object that serves as the value of this property contains two properties itself: `SubaccountId` and `InstanceId`. `SubaccountId` has as its value the ID of the subaccount instance to which the destination belongs. Depending on where the destination was found (as a subaccount destination or a service instance destination) `InstanceId` can have the value `null`, or the ID of the service instance to which the destination belongs.

- **Example:**

Sample Code

```
"owner": {
  "SubaccountId": "9acf4877-5a3d-43d2-b67d-7516efe15b11",
  "InstanceId": null
}
```

Back to [Response Structure \[page 259\]](#)

Destination Configuration

- **Key:** `destinationConfiguration`

The JSON object that represents the value of this property contains the actual properties of the destination. To learn more about the available properties, see [HTTP Destinations \[page 86\]](#).

If a destination fragment was specified in the "Find Destination" call, the value of this property is represented by the JSON object that contains the actual properties of the destination, merged with the JSON object that contains the properties of the destination fragment.

For more information on how to extend a destination with a destination fragment, see [Extending Destinations with Fragments \[page 273\]](#).

- **Example:**

Sample Code

```
"destinationConfiguration": {
  "Name": "TestBasic",
  "Type": "HTTP",
  "URL": "http://sap.com",
  "Authentication": "BasicAuthentication",
  "ProxyType": "OnPremise",
  "User": "test",
  "Password": "pass12345"
}
```

Back to [Response Structure \[page 259\]](#)

Authentication Tokens

Note

This property is only applicable to destinations that use the following authentication types: *BasicAuthentication*, *OAuth2SAMLBearerAssertion*, *OAuth2ClientCredentials*, *OAuthUserTokenExchange*, *OAuth2JWTBearer*, *OAuth2Password*, *SAMLAssertion*, *OAuth2RefreshToken*, *OAuth2AuthorizationCode*, *OAuth2TechnicalUserPropagation*.

Restriction

The section will contain an error if the `tokenServiceUrl` is a private endpoint (like *localhost*) and `ProxyType` is **Internet**, as the automation cannot be performed on the server side in this case.

Note

This section will not be present if automatic token retrieval is skipped by setting the query parameter `$skipTokenRetrieval=true`.

- **Key:** `authTokens`

The JSON array that represents the value of this property contains tokens that are required for authentication. These tokens are represented by JSON objects with these properties (expect more new properties to be added in the future):

- `type`: the type of the token.
- `value`: the actual token.
- `http_header`: JSON object containing the prepared token in the correct format. The `<key>` field contains the key of the HTTP header. The `<value>` field contains the value of the header.
- `expires_in` (only in OAuth2 destinations): The lifetime in seconds of the access token. For example, the value "3600" denotes that the access token will expire in one hour from the time the response was generated.
- `error` (optional): if the retrieval of the token fails, the value of both `type` and `value` is an empty string and this property shows an error message, explaining the problem.
- `scope` (optional) (only in OAuth2 destinations): The scopes issued with the token. The value of the scope parameter is expressed as a list of space-delimited strings. For example, `read write execute`.
- `refresh_token` (optional) (only in OAuth2 destinations): A refresh token, returned by the OAuth service. It can be used to renew the access token via [OAuth Refresh Token Authentication \[page 134\]](#).

- **Example:**

Sample Code

```
"authTokens": [  
  {  
    "type": "Basic",  
    "value": "dGVzdDpwYXNzMTIzNDU=",  
    "http_header": {  
      "key": "Authorization",  
      "value": "Basic dGVzdDpwYXNzMTIzNDU="
```

Back to [Response Structure \[page 259\]](#)

Certificates

Note

This property is only applicable to destinations that use mTLS for token retrieval and automatic token retrieval is skipped, or destinations with the following authentication types: *ClientCertificateAuthentication*, *OAuth2SAMLBearerAssertion* (when default JDK trust store is not used).

- **Key:** `certificates`
The JSON array that represents the value of this property contains the certificates, specified in the destination configuration. These certificates are represented by JSON objects with these properties (expect more new properties to be added in the future):
 - `type`
 - `content`: the encoded content of the certificate.
 - `name`: the name of the certificate, as specified in the destination configuration.
- **Example:**

Sample Code

```
"certificates": [  
  {  
    "Name": "keystore.jks",  
    "Content": "<value>",  
    "Type": "CERTIFICATE"  
  },  
  {  
    "Name": "truststore.jks",  
    "Content": "<value>",  
    "Type": "CERTIFICATE"  
  }  
]
```

Back to [Response Structure \[page 259\]](#)

Example

Example of a full response for a destination using basic authentication:

Sample Code

```
{  
  "owner": {  
    "SubaccountId": "9acf4877-5a3d-43d2-b67d-7516efe15b11",
```

```

    "InstanceId": null
  },
  "destinationConfiguration": {
    "Name": "TestBasic",
    "Type": "HTTP",
    "URL": "http://sap.com",
    "Authentication": "BasicAuthentication",
    "ProxyType": "OnPremise",
    "User": "test",
    "Password": "pass12345"
  },
  "authTokens": [
    {
      "type": "Basic",
      "value": "dGVzdDpwYXNzMTIzNDU="
      "http_header": {
        "key": "Authorization",
        "value": "Basic dGVzdDpwYXNzMTIzNDU="
      }
    }
  ]
}

```

Back to [Response Structure \[page 259\]](#)

1.1.4.2.3 Exchanging User JWTs via OAuth2UserTokenExchange Destinations

Automatic token retrieval using the `OAuth2UserTokenExchange` authentication type for HTTP destinations.

Content

[Prerequisites \[page 263\]](#)

[Automated Access Token Retrieval \[page 264\]](#)

[Scenarios \[page 264\]](#)

Prerequisites

You have configured an `OAuth2UserTokenExchange` destination. See [OAuth User Token Exchange Authentication \[page 112\]](#).

The token to be exchanged must have the `uaa.user` scope to enable the token exchange. See [SAP Authorization and Trust Management Service](#) for more details.

Back to [Content \[page 263\]](#)

Automated Access Token Retrieval

For destinations of authentication type `OAuth2UserTokenExchange`, you can use the automated token retrieval functionality via the "find destination" endpoint, see [Call the Destination Service \[page 247\]](#).

If you provide the user token exchange header with the request to the Destination service and its value is not empty, it is used instead of the `Authorization` header to specify the user and the tenant subdomain. It will be the token for which token exchange is performed.

- The header value must be a user JWT (JSON Web token) in encoded form, see [RFC 7519](#) .
- If the user token exchange header is not provided with the request to the Destination Service or it is provided, but its value is empty, the token from the `Authorization` header is used instead. In this case, the JWT in the `Authorization` header must be a user JWT in encoded form, otherwise the token exchange does not work.

For information about the response structure of this request, see ["Find Destination" Response Structure \[page 259\]](#).

Back to [Content \[page 263\]](#)

Scenarios

To achieve specific token exchange goals, you can use the following headers and values when calling the Destination service:

Goal	User Token Exchange Header	Authorization Header
Exchange a user token: <ul style="list-style-type: none">• Issued on behalf of the application provider tenant• Using a destination in the application provider tenant	Not used	The user token to be exchanged Previously retrieved by the application via exchanging the initial user token, passed to the application (to another user access token) via the client credentials of the Destination service instance (bound to the application), using the provider tenant-specific token service URL.
Exchange a user token: <ul style="list-style-type: none">• Issued on behalf of a tenant, subscribed to your application• Using a destination in the application provider tenant	<User token to be exchanged>	Access token Retrieved via the client credentials of the Destination service instance (bound to the application), using the provider tenant-specific token service URL.

Goal	User Token Exchange Header	Authorization Header
Exchange a user token: <ul style="list-style-type: none"> Issued on behalf of a tenant, subscribed to your application Using a destination in the subscriber tenant 	<User token to be exchanged>	Access token Retrieved via the client credentials of the Destination service instance (bound to the application), using the subscriber tenant-specific token service URL.

Back to [Content \[page 263\]](#)

1.1.4.2.4 Multitenancy in the Destination Service

Establish multitenancy in the Destination service using subscription-level destinations.

Concept

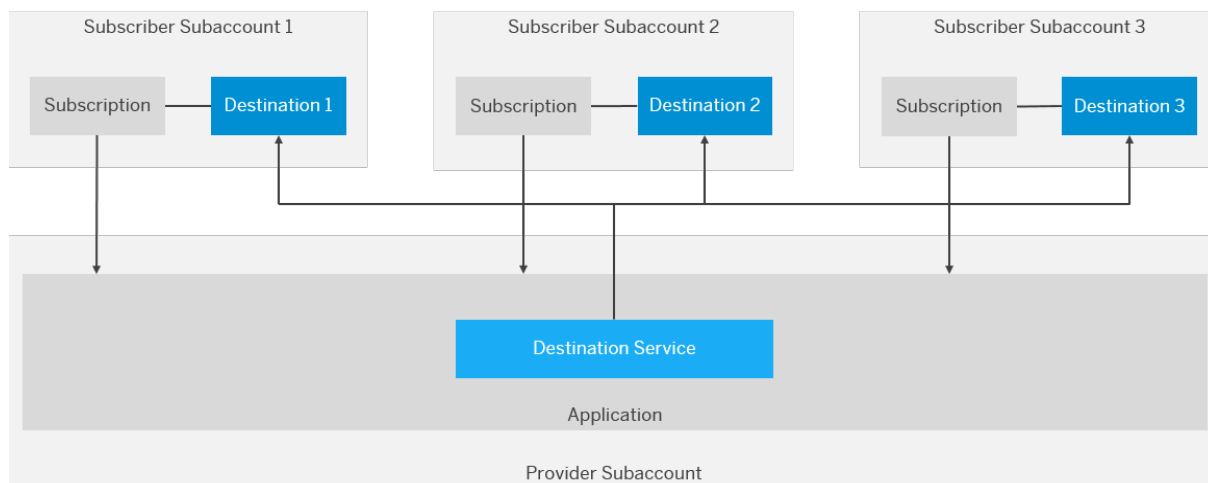
When developing a provider application (SaaS application) that consumes the Destination service, you can choose between the following destination levels:

Level	Who has Access and How?
Subaccount	Any application using <i>any</i> destination service instance in the subaccount context. This is the common level for all applications and service instances in the subaccount.
Service Instance	Any application using the concrete destination service instance in the context of the <i>provider</i> subaccount (the subaccount in which the instance is provisioned). Each service instance has its own level.
Subscription	Any application using the <i>concrete</i> destination service instance in the context of the <i>subscribed</i> subaccount. Each combination of service instance and subscriber subaccount is a unique level.

📌 Note

The term *level* is used here to represent an area or visibility scope. The higher the level, the broader is the visibility scope.

If you, as an application provider, want to create a destination that is used at runtime *only by the subscriber* and that should be *visible and accessible only to the subscriber*, you can create a subscription-level destination for each subscriber subaccount (tenant):



[Create a Subscription-Level Destination \[page 266\]](#)

[Consume a Subscription-Level Destination \[page 266\]](#)

Create a Subscription-Level Destination

1. Retrieve an OAuth token from the subscriber token service URL using the OAuth client credentials from the destination service instance, for example:

```
POST https://{subscriber subdomain}.authentication.{region host}/oauth/token
```

2. Use the retrieved token from step 1 to create (POST) a subscription-level destination in the Destination service, see *Destinations on service instance (subscription) level* in the [REST API specification](#).

[Back to Concept \[page 265\]](#)

Consume a Subscription-Level Destination

1. Retrieve an OAuth token from the subscriber token service URL using the OAuth client credentials from the destination service instance, for example:

```
POST https://{subscriber subdomain}.authentication.{region host}/oauth/token
```

2. Use the token from step 1 to retrieve (GET) the destination stored on subscription level from the Destination service via:
 - *Find a destination* in the [REST API specification](#)
 - *Destinations on service instance (subscription) level* in the [REST API specification](#)

[Back to Concept \[page 265\]](#)

Related Information

[Multitenancy in the Connectivity Service \[page 203\]](#)

1.1.4.2.5 Destination Java APIs

Use Destination service Java APIs to optimize application development in the Cloud Foundry environment.

When running your cloud application with *SAP Java Buildpack*, you can use the following Java APIs to optimize the application development:

- [ConnectivityConfiguration API \[page 267\]](#): Retrieve destination configurations.
- [AuthenticationHeaderProvider API \[page 270\]](#): Retrieve prepared authentication headers, ready to be used towards the remote target system.

Add the *Connectivity Apiext* dependency in the `pom.xml` file:

```
<dependency>
  <groupId>com.sap.cloud.connectivity.apiext</groupId>
  <artifactId>com.sap.cloud.connectivity.apiext</artifactId>
  <version>${connectivity-apiext.version}</version>
  <scope>provided</scope>
</dependency>
```

For more information on *SAP Java Buildpack*, see [Developing Java in the Cloud Foundry Environment](#).

1.1.4.2.5.1 ConnectivityConfiguration API

Use the `ConnectivityConfiguration` API to retrieve destination configurations and certificate configurations in the Cloud Foundry environment.

Overview

The `ConnectivityConfiguration` API is visible by default from the web applications hosted on SAP Java Buildpack. You can access it via a JNDI lookup.

Besides managing destination configurations, you can also allow your applications to use their own managed HTTP clients. The `ConnectivityConfiguration` API provides you with direct access to the destination configurations of your applications.

To learn how to retrieve destination configurations, follow the procedure below.

Procedure

1. To consume a connectivity configuration using JNDI, you must define the `ConnectivityConfiguration` API as a resource in the `context.xml` file.

Example of a `ConnectivityConfiguration` resource named `connectivityConfiguration`, which is described in the `context.xml` file:

src/main/webapp/META-INF/context.xml

Sample Code

```
<?xml version='1.0' encoding='utf-8'?>
<Context>
  <Resource name="connectivityConfiguration" auth="Container"
  type="com.sap.core.connectivity.api.configuration.ConnectivityConfiguration"
  factory="com.sap.core.connectivity.api.jndi.ServiceObjectFactory"/>
</Context>
```

2. You also need to enable `Connectivity ApiExt` with an environment variable and bind the appropriate services in `manifest.yml`:

manifest.yml

Sample Code

```
applications:
- ...
  env:
    USE_CONNECTIVITY_APIEXT: true
  services:
    - xsuaa-instance
    - destination-instance
    - connectivity-instance
  ...
```

3. In your servlet code, you can look up the `ConnectivityConfiguration` API from the JNDI registry as follows:

Sample Code

```
import javax.naming.Context;
import javax.naming.InitialContext;
import com.sap.core.connectivity.api.configuration.ConnectivityConfiguration;
...

// look up the connectivity configuration API "connectivityConfiguration"
Context ctx = new InitialContext();
ConnectivityConfiguration configuration = (ConnectivityConfiguration)
ctx.lookup("java:comp/env/connectivityConfiguration");
```

4. With the retrieved `ConnectivityConfiguration` API, you can read all properties of any destination defined on subscription, application, or subaccount level.

Sample Code

```
// get destination configuration for "myDestinationName"
DestinationConfiguration destConfiguration =
configuration.getConfiguration("myDestinationName");

// get a single destination property
String authenticationType =
destConfiguration.getProperty("Authentication");

// get all destination properties
Map<String, String> allDestinationProperties =
destConfiguration.getAllProperties();
```

Note

If you have two destinations with the same name, for example, one configured on subaccount level and the other on service instance/subscription level, the `getConfiguration()` method will return the destination on instance/subscription level.

The preference order is:

1. Instance/subscription level
2. Subaccount level

5. If a trust store and a key store are defined in the corresponding destination, you can access them by using the methods `getKeyStore` and `getTrustStore`.

Sample Code

```
// get destination configuration for "myDestinationName"
DestinationConfiguration destConfiguration =
configuration.getConfiguration("myDestinationName");

// get the configured keystore
KeyStore keyStore = destConfiguration.getKeyStore();

// get the configured truststore
KeyStore trustStore = destConfiguration.getTrustStore();

// create sslcontext
TrustManagerFactory tmf =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
tmf.init(trustStore);

KeyManagerFactory keyManagerFactory =
KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());

// get key store password from destination
String keyStorePassword =
destConfiguration.getProperty("KeyStorePassword");
keyManagerFactory.init(keyStore, keyStorePassword.toCharArray());

SSLContext sslContext = SSLContext.getInstance("TLSv1");
sslContext.init(keyManagerFactory.getKeyManagers(),
tmf.getTrustManagers(), null);
```

```

SSLConnectionFactory sslConnectionFactory = sslcontext.getSocketFactory();

// get the destination URL
String value = destConfiguration.getProperty("URL");
URL url = new URL(value);

// use the sslcontext for url connection
URLConnection urlConnection = url.openConnection();
((HttpsURLConnection) urlConnection).setSSLConnectionFactory(sslConnectionFactory);
urlConnection.connect();
InputStream in = urlConnection.getInputStream();
...

```

1.1.4.2.5.2 AuthenticationHeaderProvider API

Use the `AuthenticationHeaderProvider` API for applications in the Cloud Foundry environment to retrieve prepared authentication headers that are ready to be used to towards a remote target system.

Overview

The `AuthenticationHeaderProvider` API is visible by default from the web applications hosted on *SAP Java Buildpack*. You can access it via a JNDI lookup.

This API lets your applications use their own managed HTTP clients, as it provides them with automated authentication token retrieval, making it easy to implement *single sign-on* (SSO) towards the remote target.

Procedure

1. To consume the authentication header provider API using JNDI, you need to define the `AuthenticationHeaderProvider` API as a resource in the `context.xml` file. Example of an `AuthenticationHeaderProvider` resource named `myAuthHeaderProvider`, which is described in the `context.xml` file:

Sample Code

```

<?xml version='1.0' encoding='utf-8'?>

<Context>
  <Resource name="myAuthHeaderProvider" auth="Container"

type="com.sap.core.connectivity.api.authentication.AuthenticationHeaderProv
ider"

factory="com.sap.core.connectivity.api.jndi.ServiceObjectFactory"/>
</Context>

```

2. In your servlet code, you can look up the `AuthenticationHeaderProvider` API from the JNDI registry as follows:

Sample Code

```
import javax.naming.Context;
import javax.naming.InitialContext;
import
com.sap.core.connectivity.api.authentication.AuthenticationHeaderProvider;
...

// look up the connectivity authentication header provider resource called
"myAuthHeaderProvider"
Context ctx = new InitialContext();
AuthenticationHeaderProvider authHeaderProvider
= (AuthenticationHeaderProvider) ctx.lookup("java:comp/env/
myAuthHeaderProvider");
```

Single Sign-On to On-Premise Systems

The Connectivity service supports a mechanism to enable SSO using the so-called *principal propagation* authentication type of a destination configuration. To propagate the logged-in user, the application can use the `AuthenticationHeaderProvider` API to retrieve a prepared HTTP header, which then embeds in the HTTP request to the on-premise system.

Prerequisites

To connect to on-premise systems and perform single sign-on, you must bind a Connectivity service instance to the cloud application.

References

For more information, see also:

- [Principal Propagation SSO Authentication for HTTP \[page 92\]](#)
- [Create and Bind a Connectivity Service Instance \[page 206\]](#)
- [Consuming the Connectivity Service \(Java\) \(Neo environment\)](#)

Example

Sample Code

```
String proxyHost = <connectivity_service_credentials_onPremiseProxyHost>;
int proxyPort =
Integer.parseInt(<connectivity_service_credentials_onPremiseProxyPort>);
String account = SecurityContext.getAccessToken().getZoneId();

// setup the on-premise HTTP proxy
HttpClient httpClient = new DefaultHttpClient();
httpClient.getParams().setParameter(ConnRoutePNames.DEFAULT_PROXY, new
HttpHost(proxyHost, proxyPort));
```

```

// look up the connectivity authentication header provider resource called
"authHeaderProvider" (must be defined in web.xml)
Context ctx = new InitialContext();
AuthenticationHeaderProvider authHeaderProvider =
(AuthenticationHeaderProvider) ctx.lookup("java:comp/env/authHeaderProvider");

// get header for principal propagation
AuthenticationHeader principalPropagationHeader =
authHeaderProvider.getPrincipalPropagationHeader();

//insert the necessary headers in the request
HttpGet request = new HttpGet("http://virtualhost:1234");
request.addHeader(principalPropagationHeader.getName(),
principalPropagationHeader.getValue());

// execute the request
HttpResponse response = httpClient.execute(request);

```

OAuth2 SAML Bearer Assertion

The Destination service provides support for applications to use the SAML Bearer assertion flow for consuming OAuth-protected resources. In this way, applications do not need to deal with some of the complexities of OAuth and can reuse user data from existing identity providers.

Users are authenticated by using a SAML assertion against the configured and trusted OAuth token service. The SAML assertion is then used to request an access token from an OAuth token service. This access token is returned by the API and can be injected in the HTTP request to access the remote OAuth-protected resources via SSO.

→ Tip

The access tokens are cached by `AuthenticationHeaderProvider` and are auto-updated: When a token is about to expire, a new token is created shortly before the expiration of the old one.

The `AuthenticationHeaderProvider` API provides the following method to retrieve such headers:

```

List<AuthenticationHeader>
getOAuth2SAMLSAMLBearerAssertionHeaders(DestinationConfiguration
destinationConfiguration);

```

For more information, see also [Principal Propagation SSO Authentication for HTTP \[page 92\]](#).

Client Credentials

The Destination service provides support for applications to use the OAuth client credentials flow for consuming OAuth-protected resources.

The client credentials are used to request an access token from an OAuth token service.

→ Tip

The access tokens are cached by `AuthenticationHeaderProvider` and are auto-updated: When a token is about to expire, a new token is created shortly before the expiration of the old one.

The `AuthenticationHeaderProvider` API provides the following method to retrieve such headers:

```
AuthenticationHeader getOAuth2ClientCredentialsHeader (DestinationConfiguration destinationConfiguration);
```

For more information, see:

- [Principal Propagation SSO Authentication for HTTP \[page 92\]](#)
- [OAuth SAML Bearer Assertion Authentication \[page 94\]](#)
- [OAuth Client Credentials Authentication \[page 104\]](#)

Related Information

[Principal Propagation \[page 168\]](#)

1.1.4.2.6 Extending Destinations with Fragments

Use the “Find Destination” API to extend your destination with a destination fragment.

This extension involves merging the JSON object of the destination properties, together with the JSON object of the fragment properties.

ⓘ Note

If any fragment property uses the same key name as a destination property, the combined JSON object will use the value of the fragment property.

The combined JSON of the destination and fragment will be returned in the response body as the value of the “destinationConfiguration” property. Additionally, this combined JSON will be used for retrieving tokens from authorization servers, if applicable.

ⓘ Note

The combination of the destination JSON and the fragment JSON happens in the context of the “Find Destination” request. The actual destination stored in Destination service is not modified.

To apply this mechanism, you must use the following configurations:

- A Destination – with any properties

- A Destination Fragment – with any properties

Note

For more information on how to create and manage resources like destinations and destination fragments, see [SAP Business Accelerator Hub](#).

Use a Header to Specify the Destination Fragment

Header	Value	Description
X-Fragment-Name	Name of a destination fragment	Name of the destination fragment. The fragment itself must be maintained on the same level as the destination.

Caution

Only one X-Fragment-Name header may be present in the “Find Destination” request.

Example: Using Destination Fragments to Add New Properties to a Destination

Destination:

Sample Code

```
Name=destination
Type=HTTP
URL=https://xxxx.example.com
ProxyType=Internet
Authentication=NoAuthentication
```

Destination Fragment:

Sample Code

```
FragmentName=fragment
example-property=example-value
```

cURL call to send "Find Destination" request:

Sample Code

```
curl --request GET --url 'https://<destination_service_host>/destination-configuration/v1/destinations/<name_of_destination_being_extended>' \
--header 'Accept: */*' \
--header 'Authorization: Bearer <access_token>' \
--header 'X-Fragment-Name: <name_of_fragment>'
```

Response:

Sample Code

```
{
  "owner": {
    "SubaccountId": <subaccount_id>,
    "InstanceId": null
  },
  "destinationConfiguration": {
    "Name": "destination",
    "Type": "HTTP",
    "URL": "https://xxxx.example.com",
    "Authentication": "NoAuthentication",
    "ProxyType": "Internet",
    "FragmentName": "fragment",
    "example-property": "example-value"
  }
}
```

Example: Using Destination Fragments to Override Properties of a Destination

Destination:

Sample Code

```
Name=destination
Type=HTTP
URL=https://xxxx.example.com
ProxyType=Internet
Authentication=OAuth2ClientCredentials
clientId=clientId
clientSecret=secret1234
tokenServiceURL=https://authserver1.example.com/oauth/token/
```

Destination Fragment:

Sample Code

```
FragmentName=fragment  
clientId=clientId-2  
clientSecret=secret2345  
tokenServiceURL=https://authserver2.example.com/oauth/token/
```

cURL call to send "Find Destination" request:

Sample Code

```
curl --request GET \  
--url 'https://<destination_service_host>/destination-configuration/v1/  
destinations/<name_of_destination_being_extended>' \  
--header 'Accept: */*' \  
--header 'Authorization: Bearer <access_token>' \  
--header 'X-Fragment-Name: <name_of_fragment>'
```

Response:

Sample Code

```
{  
  "owner": {  
    "SubaccountId": <subaccount_id>,  
    "InstanceId": null  
  },  
  "destinationConfiguration": {  
    "Name": "destination",  
    "Type": "HTTP",  
    "URL": "https://xxxx.example.com",  
    "Authentication": "OAuth2ClientCredentials",  
    "ProxyType": "Internet",  
    "FragmentName": "fragment",  
    "clientId": "clientId-2",  
    "clientSecret": "secret2345",  
    "tokenServiceURL": "https://authserver2.example.com/oauth/token/"  
  },  
  "authTokens": [  
    {  
      "type": "Bearer",  
      "value": "eyJhbGciOiJIUzUzIiwiaWF0Ijoi...",  
      "http_header": {  
        "key": "Authorization",  
        "value": "Bearer eyJhbGciOiJIUzUzIiwiaWF0Ijoi..."  
      }  
    }  
  ]  
}
```

Related Information

[Calling the Destination Service REST API \[page 256\]](#)



["Find Destination" Response Structure \[page 259\]](#)

1.1.4.3 Invoking ABAP Function Modules via RFC

Call a remote-enabled function module in an on-premise or cloud ABAP server from your Cloud Foundry application, using the RFC protocol.

Find the tasks and prerequisites that are required to consume an ABAP function module via RFC, using the Java Connector (JCo) API as a built-in feature of SAP BTP.

Tasks

Task Type	Task
 Operator	Prerequisites [page 277]
 Operator and/or Developer	About JCo [page 278] Installation Prerequisites for JCo Applications [page 278] Consume Connectivity via RFC [page 279] Restrictions [page 279]

Prerequisites



Before you can use RFC communication for an SAP BTP application, you must configure:

- A destination on SAP BTP to use RFC.
For more information, see [RFC Destinations \[page 157\]](#).
- (Only for on-premise backend systems) RFC connectivity between a backend system and the application.
To do this, you must install the [Cloud Connector \[page 343\]](#) in your internal network and configure it to expose a remote-enabled function module in an ABAP system.

For more information, see [Initial Configuration \(RFC\) \[page 398\]](#) and [Configure Access Control \(RFC\) \[page 465\]](#).

Back to [Tasks \[page 277\]](#)

About JCo



To learn in detail about the SAP JCo API, see the JCo 3.0 documentation on [SAP Support Portal](#).

Note

Some sections of this documentation are not applicable to SAP BTP:

- **Architecture:** CPIC is only used in the last mile from your Cloud Connector to an *on-premise* ABAP backend. From SAP BTP to the Cloud Connector, TLS-protected communication is used.
- **Installation:** SAP BTP runtimes already include all required artifacts.
- **Customizing and Integration:** On SAP BTP, the integration is already done by the runtime. You can concentrate on your business application logic.
- **Server Programming:** The programming model of JCo on SAP BTP does not include server-side RFC communication.
- **IDoc Support for External Java Applications:** Currently, there is no IDocLibrary for JCo available on SAP BTP

Back to [Tasks \[page 277\]](#)

Installation Prerequisites for JCo Applications



- For connections to an *on-premise* ABAP backend, you have downloaded the Cloud Connector installation archive from [SAP Development Tools for Eclipse](#) and connected the Cloud Connector to your subaccount.
- You have downloaded and set up your Eclipse IDE and the [Eclipse Tools for Cloud Foundry](#).
- You have downloaded the Cloud Foundry CLI, see [Tools](#).

Back to [Tasks \[page 277\]](#)

Consuming Connectivity via RFC



You can call a service from a fenced customer network using an application that consumes a remote-enabled function module.

Invoking function modules via RFC is enabled by a JCo API that is comparable to the one available in SAP NetWeaver Application Server Java (version 7.10), and in JCo standalone 3.0. If you are an experienced JCo developer, you can easily develop a Web application using JCo: you simply consume the APIs like you do in other Java environments. Restrictions that apply in the cloud environment are mentioned in the **Restrictions** section below.

Find a sample Web application in [Invoke ABAP Function Modules in On-Premise ABAP Systems \[page 280\]](#).

Back to [Tasks \[page 277\]](#)

Restrictions



- The supported **runtime environment** is SAP Java Buildpack as of version 1.8.0.

Note

You must use the Tomcat or TomEE runtime offered by the build pack to make JCo work correctly. You cannot bring a container of your own.

- Your application must not bundle the JCo 3.1 standalone Java archives nor the native library. JCo is already embedded properly in the build pack.
- **JCoServer** functionality cannot be used within SAP BTP.
- **Environment embedding**, such as offered by JCo standalone 3.1, is not possible. This is, however, similar to SAP NetWeaver AS Java.
- A **stateful sequence of function module invocations** must be done in a single HTTP request/response cycle.
- **Logon authentication** only supports user/password credentials (basic authentication) and principal propagation. See [Authentication to the On-Premise System \[page 223\]](#).
- The supported set of **configuration properties** is restricted. For details, see [RFC Destinations \[page 157\]](#).

Back to [Tasks \[page 277\]](#)

Related Information

- [Use Cases \[page 280\]](#)
- [Developing Java in the Cloud Foundry Environment](#)
- [SAP Java Connector](#)

1.1.4.3.1 Use Cases

Find instructions for typical RFC end-to-end scenarios that use the Connectivity service and/or the Destination service (Cloud Foundry environment).









Invoke ABAP Function Modules in On-Premise ABAP Systems [page 280]	Call a function module in an on-premise ABAP system via RFC, using a sample Web application (Cloud Foundry environment).
Invoke ABAP Function Modules in Cloud ABAP Systems [page 305]	Call a function module in a cloud ABAP system via RFC, using a sample Web application (Cloud Foundry environment).
Multitenancy for JCo Applications (Advanced) [page 323]	Learn about the required steps to make your Cloud Foundry JCo application tenant-aware.
Configure Principal Propagation for RFC [page 334]	Enable single sign-on (SSO) via RFC by forwarding the identity of cloud users from the Cloud Foundry environment to an on-premise system.

1.1.4.3.1.1 Invoke ABAP Function Modules in On-Premise ABAP Systems

Call a function module in an on-premise ABAP system via RFC, using a sample Web application (Cloud Foundry environment).

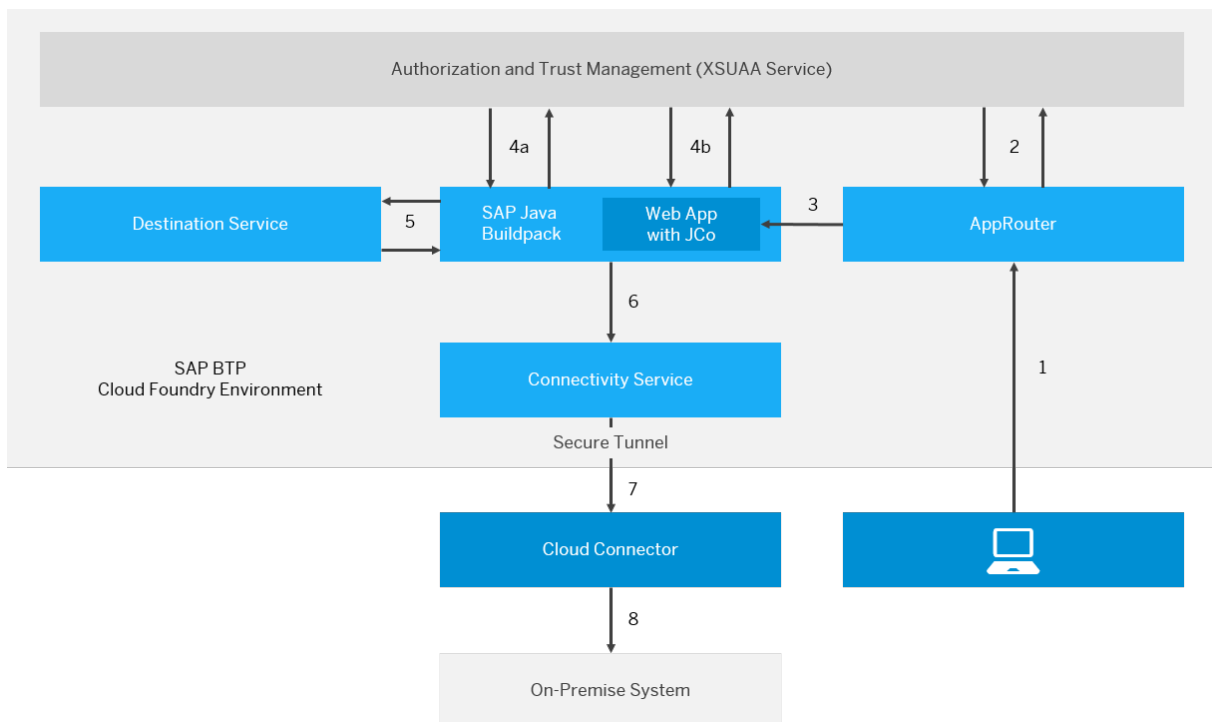
This scenario performs a remote function call to invoke an ABAP function module, by using the Connectivity service and the Destination service in the Cloud Foundry environment, as well as a Cloud Foundry application router.

Tasks

Task Type	Task
 Operator and/or Developer	Scenario Overview [page 281] Connectivity User Roles [page 283] Installation Prerequisites [page 284] Used Values [page 283]
 Developer	Develop a Sample Web Application [page 284]
 Operator and/or Developer	Create and Bind Service Instances [page 289]
 Developer	Deploy the Application [page 292]
 Operator	Configure Roles and Trust [page 294]
 Operator and/or Developer	Set Up an Application Router [page 295]
 Operator	Configure the RFC Destination [page 299] Configure the Cloud Connector [page 300]
 Operator and/or Developer	Monitoring Your Web Application [page 304] (Optional)

Scenario Overview

Control Flow for Using the Java Connector (JCo) with Basic Authentication



Process Steps:

1. Call through AppRouter (entry point for business applications).

Note

AppRouter is only required if you want to use multitenancy or perform user-specific service calls. In all other cases, JCo uses [cloud-security-xsuaa-integration](#) with `ClientCredentialFlow`.

2. Redirect to XSUAA for login. JSON Web Token (JWT1) is sent to AppRouter and cached there.
3. AppRouter calls Web app and sends JWT1 with credentials.
4. Buildpack/XSUAA interaction:
 1. Buildpack requests JWT2 to access the Destination service instance (JCo call).
 2. Buildpack requests JWT3 to access the Connectivity service instance.
5. Buildpack requests destination configuration (JWT2).
6. Buildpack sends request to the Connectivity service instance (with JWT3 and Authorization Header).
7. Connectivity service forwards request to the Cloud Connector.
8. Cloud Connector sends request to on-premise system.

Since token exchanges are handled by the buildpack, you must only create and bind the service instances, see [Create and Bind Service Instances \[page 289\]](#).

Back to [Tasks \[page 280\]](#)

Used Values

This scenario uses:

- A subaccount in region **Europe (Frankfurt)**, for which the API endpoint is `api.cf.eu10.hana.ondemand.com`.
- The application name `jco-demo-<subaccount name>`, where `<subaccount name>` is the subdomain name of the subaccount. For this example, we use `p1234`.

Back to [Tasks \[page 280\]](#)

Connectivity User Roles

Different user roles are involved in the cloud to on-premise connectivity end-to-end scenario. The particular steps for the relevant roles are described below:

IT Administrator

Sets up and configures the Cloud Connector. Scenario steps:

1. Downloads the Cloud Connector from <https://tools.hana.ondemand.com/#cloud>
2. Installs the Cloud Connector.
3. Establishes a TLS tunnel from the connector to an SAP BTP subaccount.
4. Configures the exposed back-end systems and resources.

Application Developer

Develops Web applications using destinations. Scenario steps:

1. Installs Eclipse IDE, the Cloud Foundry Plugin for Eclipse and the Cloud Foundry CLI.
2. Develops a Java EE application using the JCo APIs.
3. Configures connectivity destinations via the SAP BTP cockpit.
4. Deploys and tests the Java EE application on SAP BTP.

Account Operator

Deploys Web applications, creates application routers, creates and binds service instances, conducts tests. Scenario steps:

1. Obtains a ready Java EE application WAR file.
2. Deploys an application router with respective routes to the application.
3. Creates an XSUAA service instance and binds it to the router.
4. Deploys the Java EE application to an SAP BTP subaccount.
5. Creates a Connectivity service and Destination service instance, and binds them to the application.
6. Creates and manages roles and role collections.

Back to [Tasks \[page 280\]](#)

Installation Prerequisites

- You have downloaded the Cloud Connector installation archive from [SAP Development Tools for Eclipse](#) and connected the Cloud Connector to your subaccount.
- You have downloaded and set up your Eclipse IDE and the [Eclipse Tools for Cloud Foundry](#) .
- You have downloaded the Cloud Foundry CLI, see [Tools](#).

Back to [Tasks \[page 280\]](#)

Next Steps

- [Develop a Sample Web Application \[page 284\]](#)
- [Create and Bind Service Instances \[page 289\]](#)
- [Deploy the Application \[page 292\]](#)
- [Configure Roles and Trust \[page 294\]](#)
- [Set Up an Application Router \[page 295\]](#)
- [Configure the RFC Destination \[page 299\]](#)
- [Configure the Cloud Connector \[page 300\]](#)
- [Monitoring Your Web Application \[page 304\] \(Optional\)](#)

Related Information

[Multitenancy for JCo Applications \(Advanced\) \[page 323\]](#)

1.1.4.3.1.1.1 Develop a Sample Web Application

Create a Web application to call an ABAP function module via RFC.

Steps

1. [Create a Dynamic Web Project \[page 285\]](#)
2. [Include JCo Dependencies \[page 286\]](#)
3. [Create a Sample Servlet \[page 287\]](#)

Create a Dynamic Web Project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. On the *Project Explorer* view, choose ► *New* ► *Dynamic Web Project* ▾ in the context menu.
3. Enter **jco_demo** as the project name.
4. In the *Target Runtime* pane, select **Cloud Foundry** . If it is not yet in the list of available runtimes, choose *New Runtime* and select it from there.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish*.

New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location

Use default location

Location:

Target runtime

Dynamic web module version

Configuration

A good starting point for working with Cloud Foundry runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

Add project to an EAR

EAR project name:

Working sets

Add project to working sets

Working sets:

Back to [Steps \[page 284\]](#)

Include JCo Dependencies

To use JCo functionality seamlessly at compile time in Eclipse, you must include the JCo dependencies into your web project. Therefore, you must convert it into a maven project.

1. In the *Project Explorer* view, right-click on the project `jco-demo` and choose **Configure > Convert to Maven Project**.
2. In the dialog window, leave the default settings unchanged and choose *Finish*.
3. Open the `pom.xml` file and include the following dependency:

```
<dependencies>
  <dependency>
    <groupId>com.sap.cloud</groupId>
    <artifactId>neo-java-web-api</artifactId>
    <version>[3.71.8,4.0.0)</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

Back to [Steps \[page 284\]](#)

Create a Sample Servlet

1. From the `jco_demo` project node, choose **New > Servlet** in the context menu.
2. Enter `com.sap.demo.jco` as the `<>` and `ConnectivityRFCExampleJava` as the `<Class name>`. Choose *Next*.
3. Choose *Finish* to create the servlet and open it in the Java editor.
4. Replace the entire servlet class to make use of the JCo API. The JCo API is visible by default for cloud applications. You do not need to add it explicitly to the application class path.

Sample Code

```
package com.sap.demo.jco;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.sap.conn.jco.AbapException;
import com.sap.conn.jco.JCoDestination;
import com.sap.conn.jco.JCoDestinationManager;
import com.sap.conn.jco.JCoException;
import com.sap.conn.jco.JCoFunction;
import com.sap.conn.jco.JCoParameterList;
import com.sap.conn.jco.JCoRepository;

/**
 * Sample application that uses the Connectivity
 *                               service. In particular, it is
 * making use of the capability to invoke a function module in an ABAP
system
 * via RFC
 */
```

```

*
* Note: The JCo APIs are available under <code>com.sap.conn.jco</code>.
*/
@WebServlet("/ConnectivityRFCEExample/*")
public class ConnectivityRFCEExample extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        PrintWriter responseWriter = response.getWriter();
        try {
            // access the RFC Destination "JCoDemoSystem"
            JCoDestination destination =
JCoDestinationManager.getDestination("JCoDemoSystem");

            // make an invocation of STFC_CONNECTION in the backend;
            JCoRepository repo = destination.getRepository();
            JCoFunction stfcConnection =
repo.getFunction("STFC_CONNECTION");

            JCoParameterList imports =
stfcConnection.getImportParameterList();
            imports.setValue("REQUTEXT", "SAP BTP Connectivity runs with
JCo");

            stfcConnection.execute(destination);
            JCoParameterList exports =
stfcConnection.getExportParameterList();
            String echotext = exports.getString("ECHOTEXT");
            String resptext = exports.getString("RESPTEXT");
            response.addHeader("Content-type", "text/html");
            responseWriter.println("<html><body>");
            responseWriter.println("<h1>Executed STFC_CONNECTION in system
JCoDemoSystem</h1>");
            responseWriter.println("<p>Export parameter ECHOTEXT of
STFC_CONNECTION:<br>");
            responseWriter.println(echotext);
            responseWriter.println("<p>Export parameter RESPTEXT of
STFC_CONNECTION:<br>");
            responseWriter.println(resptext);
            responseWriter.println("</body></html>");
        } catch (AbapException ae) {
            // just for completeness: As this function module does not
have an exception
            // in its signature, this exception cannot occur. But you
should always
            // take care of AbapExceptions
        } catch (JCoException e) {
            response.addHeader("Content-type", "text/html");
            responseWriter.println("<html><body>");
            responseWriter
                .println("<h1>Exception occurred while executing
STFC_CONNECTION in system JCoDemoSystem</h1>");
            responseWriter.println("<pre>");
            e.printStackTrace(responseWriter);
            responseWriter.println("</pre>");
            responseWriter.println("</body></html>");
        }
    }
}

```

5. Save the Java editor and make sure that the project compiles without errors.

Back to [Steps \[page 284\]](#)

Next Steps

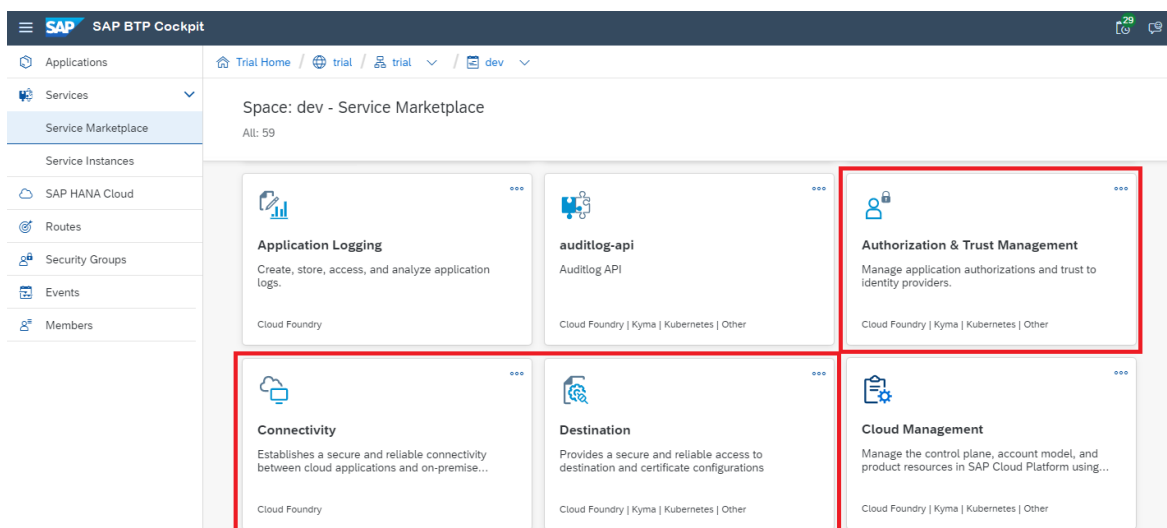
- [Create and Bind Service Instances](#) [page 289]
- [Deploy the Application](#) [page 292]
- [Configure Roles and Trust](#) [page 294]
- [Set Up an Application Router](#) [page 295]
- [Configure the RFC Destination](#) [page 299]
- [Configure the Cloud Connector](#) [page 300]
- [Monitoring Your Web Application](#) [page 304] (Optional)

1.1.4.3.1.1.2 Create and Bind Service Instances

You must create and bind several service instances, before you can use your application.

Procedure

1. Logon to the cloud cockpit and choose your subaccount.
2. Choose the space where you want to deploy your demo application.
3. Choose *Service Marketplace* and find these 3 services:
 - *Connectivity*
 - *Destination*
 - *Authorization & Trust Management* (XSUAA)



4. Create and bind a service instance for each of these services.
 - [Connectivity service](#) [page 290]
 - [Destination service](#) [page 290]

- [Authorization & Trust Management \(XSUAA service\) \[page 291\]](#)

Connectivity Service

1. Choose [Connectivity](#) > [Create Instance](#) .
2. Insert an instance name (for example, `connectivity_jco`) and choose [Create Instance](#) .

New Instance

1 Basic Info

2 Parameters

3 Review

Enter basic info for your service instance

Service: *
Connectivity

Service Plan: *
lite

Instance Name: * [i](#)
<instance_name>

Next > **Create Instance** Cancel

[Back to Procedure \[page 289\]](#)

Destination Service

1. Choose **Destination** > **Create Instance**.
2. Insert an instance name (for example, `destination_jco`) and choose **Create Instance**.

Back to [Procedure \[page 289\]](#)

Authorization & Trust Management (XSUAA Service)

1. Choose **Authorization & Trust Management** > **Create Instance**.
2. Select **Service Plan** `application`.
3. Enter an **Instance Name** and choose **Next**.

Note

The instance name must match the one defined in the manifest file.

4. In the next tab **Parameters**, insert the following as a JSON file:

Sample Code

```
{
  "xsappname" : "jco-demo-p1234",
  "tenant-mode": "dedicated",
  "scopes": [
    {
      "name": "$XSAPPNAME.all",
      "description": "all"
    }
  ],
  "role-templates": [
    {
      "name": "all",
      "description": "all",
      "scope-references": [
        "$XSAPPNAME.all"
      ]
    }
  ]
}
```

5. Go to tab **Review** and choose **Create Instance**.

Back to [Procedure \[page 289\]](#)

Next Steps

- [Deploy the Application \[page 292\]](#)
- [Configure Roles and Trust \[page 294\]](#)
- [Set Up an Application Router \[page 295\]](#)

- [Configure the RFC Destination \[page 299\]](#)
- [Configure the Cloud Connector \[page 300\]](#)
- [Monitoring Your Web Application \[page 304\]](#) (Optional)

1.1.4.3.1.1.3 Deploy the Application

Deploy your Cloud Foundry application to call an ABAP function module via RFC.

Prerequisites

You have created and bound the required service instances, see [Create and Bind Service Instances \[page 289\]](#).

Procedure

1. To deploy your Web application, you can use the following two alternative procedures:
 - [Deploying from the Eclipse IDE](#)
 - Deploying from the CLI, see [Developing Java in the Cloud Foundry Environment](#)
2. In the following, we publish it with the CLI.
3. To do this, create a `manifest.yml` file. The key parameter is `USE_JCO: true`, which must be set to include JCo into the buildpack during deployment.

Note

JCo supports the usage of X.509 secrets for communication with the Destination/Connectivity service. If you want to use it, you must specify the binding accordingly.

For more information, see [Binding Parameters of SAP Authorization and Trust Management Service](#).

manifest.yml

Sample Code

```
---
applications:
- name: jco-demo-p1234
  buildpacks:
  - sap_java_buildpack
  env:
    USE_JCO: true
    # This is necessary only if more than one instance is bound
    xsuaa_connectivity_instance_name: "xsuaa_jco"
    connectivity_instance_name: "connectivity_jco"
    destination_instance_name: "destination_jco"
```

```
services:
  - xsuaa_jco
  - connectivity_jco
  - destination_jco
```

⚠ Caution

The client libraries (java-security, spring-xsuaa, and container security api for node.js as of version 3.0.6) have been updated. When using these libraries, setting the parameter `SAP_JWT_TRUST_ACL` has become obsolete. This update comes with a change regarding scopes:

- For a business application A calling an application B, it is now mandatory that application B grants at least one scope to the calling business application A.
- Business application A must accept these granted scopes or authorities as part of the application security descriptor.

You can grant scopes using the `xs-security.json` file.

For more information, see [Application Security Descriptor Configuration Syntax](#), specifically the sections *Referencing the Application* and *Authorities*.

📌 Note

If you have more than one instance of those three services bound to your application, you must specify which one JCo should use with the respective env parameters:

- `xsuaa_connectivity_instance_name`
- `connectivity_instance_name`
- `destination_instance_name`.

4. In Eclipse, right-click on the project and navigate to **Export > WAR file**.
5. Choose a destination by pressing the *Browse...* button next to the `manifest.yml` you created before, for example as `jcodemo.war`.
6. Leave the other default settings unchanged and choose *Finish* to export the WAR file.
7. Perform a CLI login via `cf login -a api.cf.eu10.hana.ondemand.com -u <your_email_address>` (password, org and space are prompted after a successful login).
8. Push the application with `cf push -f manifest.yml -p jcodemo.war`.
9. Now, the application should be deployed successfully.

Next Steps

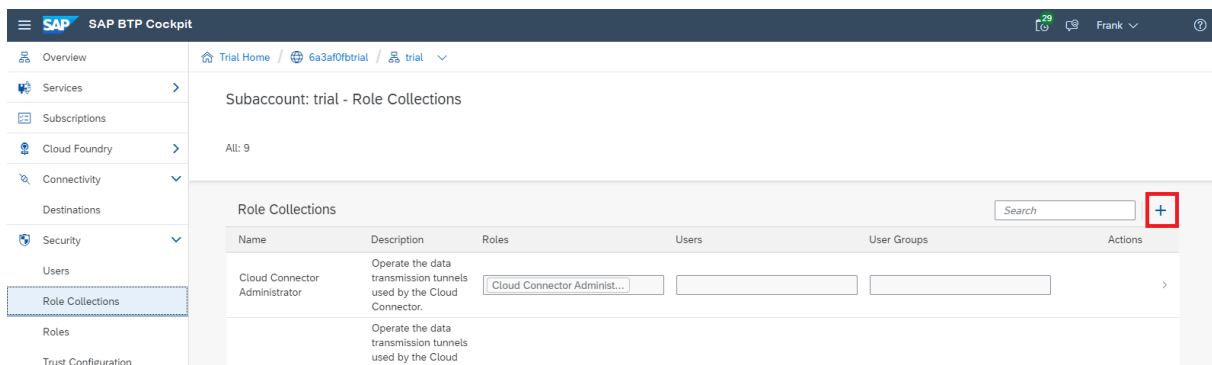
- [Configure Roles and Trust \[page 294\]](#)
- [Set Up an Application Router \[page 295\]](#)
- [Configure the RFC Destination \[page 299\]](#)
- [Configure the Cloud Connector \[page 300\]](#)

- [Monitoring Your Web Application \[page 304\]](#) (Optional)

1.1.4.3.1.1.4 Configure Roles and Trust

Configure a role that enables your user to access your Web application.

To add and assign roles, navigate to the subaccount view of the cloud cockpit and choose **Security > Role Collections**.



1. Create a new role collection with the name **a11**.
2. From the subaccount menu, choose *Trust Configuration*.
3. If you don't have a trust configuration, follow the steps in [Manually Establish Trust and Federation Between SAP Authorization and Trust Management Service and Identity Authentication](#).
4. Click on the IdP name of your choice.
5. Type in your e-mail address and choose *Show Assignments*.
6. If your user has not yet been added to the SAP ID service, you see following popup. In this case, add your user now.

Confirmation

To see and assign role collections, you must first add <user>@sap.com as a user of identity provider SAP ID Service.

Add User **Cancel**

7. You should now be able to click *Assign Role Collection*. Choose role collection **a11** and assign it.

Next Steps

- [Set Up an Application Router \[page 295\]](#)
- [Configure the RFC Destination \[page 299\]](#)
- [Configure the Cloud Connector \[page 300\]](#)
- [Monitoring Your Web Application \[page 304\]](#) (Optional)

Related Information

[Working with Role Collections](#)

1.1.4.3.1.1.5 Set Up an Application Router

For authentication purposes, configure and deploy an application router for your test application.

Note

AppRouter is only required if you want to use multitenancy or perform user-specific service calls. In all other cases, JCo uses [cloud-security-xsuaa-integration](#) with `ClientCredentialFlow`.

1. To set up an application router, follow the steps in [Application Router](#) or use the demo file [approuter.zip](#) ([download](#)).
2. For deployment, you need a manifest file, similar to this one:

Sample Code

```
---
applications:
- name:approuter-jco-demo-p1234
  path: ./
  buildpacks:
  - nodejs_buildpack
  memory: 120M
  routes:
  - route:approuter-jco-demo-p1234.cfapps.eu10.hana.ondemand.com
  env:
    NODE_TLS_REJECT_UNAUTHORIZED: 0
    destinations: >
    [
      { "name": "dest-to-example", "url" : "https://jco-
demo-p1234.cfapps.eu10.hana.ondemand.com/ConnectivityRFCEExample",
"forwardAuthToken": true }
    ]
  services:
  - xsuaa_jco
```

Note

- The routes and destination URLs need to fit your test application.
- In this example, we already bound our XSUAA instance to the application router. Alternatively, you could also do this via the cloud cockpit.

3. Push the approuter with `cf push -f manifest.yml -p approuter.zip`.
4. To navigate to the `approuter` application in the cloud cockpit, choose [▶ <your_space> ▶ Applications ▶ <your application> ▶ Overview](#).

5. When choosing the application route, you are requested to login. Provide the credentials known by the IdP you configured in *Roles & Trust*.
6. After successful login, you are routed to the test application which is then executed.
7. If the application issues an exception, saying that the `JCoDemoSystem` destination has not yet been specified, you must configure the `JCoDemoSystem` destination first.

```
Exception occurred while executing STFC_CONNECTION in system JCoDemoSystem
com.sap.conn.jco.JCoException: (106) JCO_ERROR_RESOURCE: Destination
JCoDemoSystem does not exist
    at
    com.sap.conn.jco.rt.DefaultDestinationManager.update(DefaultDestinationManager
.java:223)
    at
    com.sap.conn.jco.rt.DefaultDestinationManager.searchDestination(DefaultDestina
tionManager.java:377)
    at
    com.sap.conn.jco.rt.DefaultDestinationManager.getDestinationInstance(DefaultDe
stinationManager.java:96)
    at
    com.sap.conn.jco.JCoDestinationManager.getDestination(JCoDestinationManager.ja
va:52)
    at
    com.sap.demo.jco.ConnectivityRFCEExample.doGet(ConnectivityRFCEExample.java:47)
    .... (cut rest of the call stack)
```

Note

Make sure you **don't** include this dependency

```
<dependency>
  <groupId>com.sap.cloud.security</groupId>
  <artifactId>java-security</artifactId>
</dependency>
```

or any of its dependencies such as `java-api` with scope `compile` directly or transitively with any other jar.

Calling JCo APIs from Newly Created Threads

If you are using an Application Router and it is mandatory for you to call JCo APIs from a different thread than the one which is executing your servlet function, make sure the thread local information of the [cloud-security-xsuaa-integration API](#), used by JCo internally, is set again within your newly created thread.

To do this, add the following dependency to your project:

```
<dependency>
  <groupId>com.sap.cloud.security</groupId>
  <artifactId>java-api</artifactId>
  <version>2.7.7</version>
  <scope>provided</scope>
</dependency>
```

Adjust your code from the step [Develop a Sample Web Application \[page 284\]](#) in the following way:

Sample Code

```
package com.sap.demo.jco;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.sap.cloud.security.token.SecurityContext;
import com.sap.cloud.security.token.Token;
import com.sap.conn.jco.AbapException;
import com.sap.conn.jco.JCoDestination;
import com.sap.conn.jco.JCoDestinationManager;
import com.sap.conn.jco.JCoException;
import com.sap.conn.jco.JCoFunction;
import com.sap.conn.jco.JCoParameterList;
import com.sap.conn.jco.JCoRepository;
/**
 *
 * Sample application that uses the connectivity service. In particular, it is
 * making use of the capability to invoke a function module in an ABAP system
 * via RFC
 *
 * Note: The JCo APIs are available under <code>com.sap.conn.jco</code>.
 */
@WebServlet("/ConnectivityRFCEXample/*")
public class ConnectivityRFCEXample extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        PrintWriter responseWriter = response.getWriter();
        // access the token from the thread which is executing the servlet
        Token token = SecurityContext.getToken();
        Thread runThread = new Thread(() -> {
            // set the information in the newly created thread
            SecurityContext.setToken(token);
            try {
                // access the RFC Destination "JCoDemoSystem"
                JCoDestination destination =
JCoDestinationManager.getDestination("JCoDemoSystem_Normal");
                // make an invocation of STFC_CONNECTION in the backend
```



```

        JCoRepository repo = destination.getRepository();
        JCoFunction stfcConnection =
repo.getFunction("STFC_CONNECTION");
        JCoParameterList imports =
stfcConnection.getImportParameterList();
        imports.setValue("REQUTEXT", "SAP BTP Connectivity runs with
JCo");

        stfcConnection.execute(destination);
        JCoParameterList exports =
stfcConnection.getExportParameterList();
        String echotext = exports.getString("ECHOTEXT");
        String resptext = exports.getString("RESPTEXT");
        response.addHeader("Content-type", "text/html");
        responseWriter.println("<html><body>");
        responseWriter.println("<h1>Executed STFC_CONNECTION in
system JCoDemoSystem</h1>");
        responseWriter.println("<p>Export parameter ECHOTEXT of
STFC_CONNECTION:<br>");
        responseWriter.println(echotext);
        responseWriter.println("<p>Export parameter RESPTEXT of
STFC_CONNECTION:<br>");
        responseWriter.println(resptext);
        responseWriter.println("</body></html>");
    } catch (AbapException ae) {
        // just for completeness: As this function module does not
have an exception
        // in its signature, this exception cannot occur. But you
should always
        // take care of AbapExceptions
    } catch (JCoException e) {
        response.addHeader("Content-type", "text/html");
        responseWriter.println("<html><body>");
        responseWriter
            .println("<h1>Exception occurred while executing
STFC_CONNECTION in system JCoDemoSystem</h1>");
        responseWriter.println("<pre>");
        e.printStackTrace(responseWriter);
        responseWriter.println("</pre>");
        responseWriter.println("</body></html>");
    } finally {
        // after execution clear the token again
        SecurityContext.clearToken();
    }
}
});
runThread.start();
// wait to be finished
try {
    runThread.join();
} catch (InterruptedException e) {
    e.printStackTrace(responseWriter);
}
}
}
}

```

📌 Note

If you want to use [thread pools](#), make sure that in your thread pool implementation this information is set correctly in the thread which is about to be (re)used, and removed as soon as the thread is put back into the pool.

Next Steps

- [Configure the RFC Destination \[page 299\]](#)
- [Configure the Cloud Connector \[page 300\]](#)
- [Monitoring Your Web Application \[page 304\]](#) (Optional)

1.1.4.3.1.1.6 Configure the RFC Destination

Configure an RFC destination on SAP BTP that you can use in your Web application to call the on-premise ABAP system.

To configure the destination, you must use a virtual application server host name (**abapserver.hana.cloud**) and a virtual system number (**42**) that you will expose later in the Cloud Connector. Alternatively, you could use a load balancing configuration with a message server host and a system ID.

1. Create a `.properties` file with the following settings:

```
Name=JCoDemoSystem
Type=RFC
jco.client.ashost=abapserver.hana.cloud
jco.client.sysnr=42
jco.destination.proxy_type=OnPremise
jco.client.user=<DEMOUSER>
jco.client.passwd=<Password>
jco.client.client=000
jco.client.lang=EN
jco.destination.pool_capacity=5
```

2. Go to your subaccount in the cloud cockpit.
 1. From the subaccount menu, choose **Connectivity** > **Destinations** > **Import Destination** and upload this file.
 2. Alternatively, you can create a destination for the service instance **destination_jco** to make it visible only for this instance. To do this, go to **<your space>** > **Services** > **Service Instances** > **<your destination_instance>** and choose **Destinations**.
3. Call again the URL that references the cloud application in the Web browser. The Web application should now return a different exception:

```
Exception occurred while executing STFC_CONNECTION in system JCoDemoSystem
com.sap.conn.jco.JCoException: (102) JCO_ERROR_COMMUNICATION: Opening
connection to backend failed: Opening connection to
abapserver.hana.cloud:sapgw42 denied. Expose the system in your Cloud
Connector in case it was a valid request.
    at
com.sap.conn.jco.rt.MiddlewareJavaRfc.generateJCoException(MiddlewareJavaRfc.j
ava:487)
    at
com.sap.conn.jco.rt.MiddlewareJavaRfc$JavaRfcClient.connect(MiddlewareJavaRfc.
java:1216)
        at com.sap.conn.jco.rt.ClientConnection.connect(ClientConnection.java:700)
        at
com.sap.conn.jco.rt.RepositoryConnection.connect(RepositoryConnection.java:72)
        at com.sap.conn.jco.rt.PoolingFactory.init(PoolingFactory.java:113)
```

```
at
com.sap.conn.jco.rt.ConnectionManager.createFactory(ConnectionManager.java:426)
)
at
com.sap.conn.jco.rt.DefaultConnectionManager.createFactory(DefaultConnectionMa
nager.java:46)
at
com.sap.conn.jco.rt.ConnectionManager.getFactory(ConnectionManager.java:376)
at
com.sap.conn.jco.rt.RfcDestination.getSystemID(RfcDestination.java:1101)
..... (cut rest of the call stack)
```

4. This means that the Cloud Connector denied opening a connection to this system. As a next step, you must configure the system in your installed Cloud Connector.

Next Steps

- [Configure the Cloud Connector \[page 300\]](#)
- [Monitoring Your Web Application \[page 304\]](#) (Optional)

Related Information

[Target System Configuration \[page 163\]](#)

1.1.4.3.1.1.7 Configure the Cloud Connector

Configure the system mapping and the function module in the Cloud Connector.

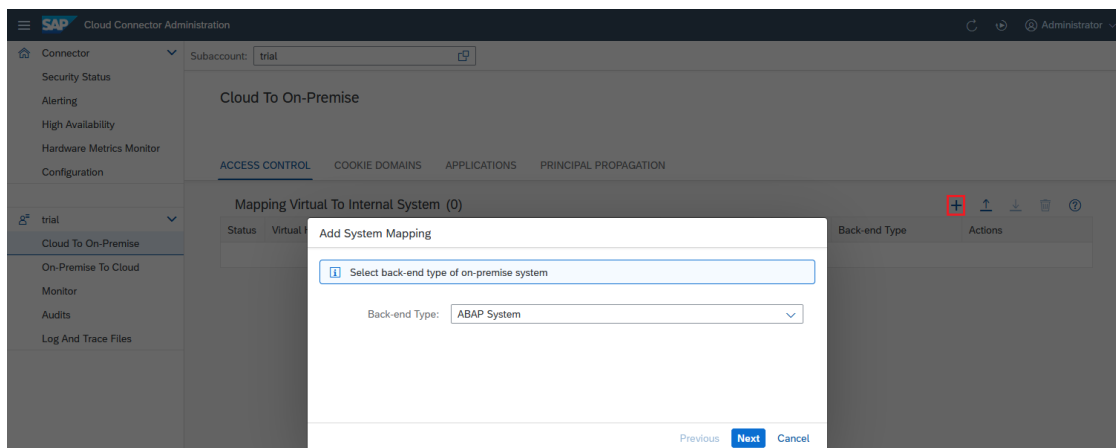
Steps

1. [Configure Host Mapping \[page 300\]](#)
2. [Configure the Function Module \[page 303\]](#)

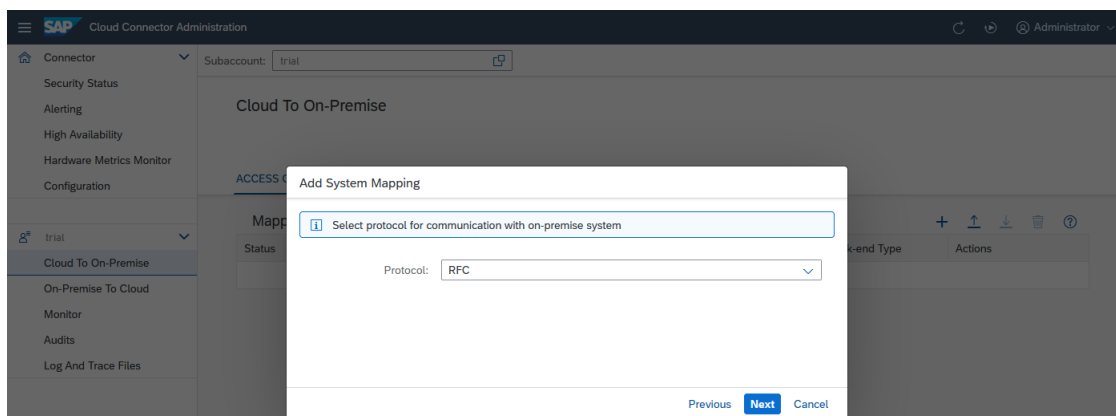
Configure Host Mapping

The Cloud Connector only allows access to trusted backend systems. To configure this, follow the steps below:

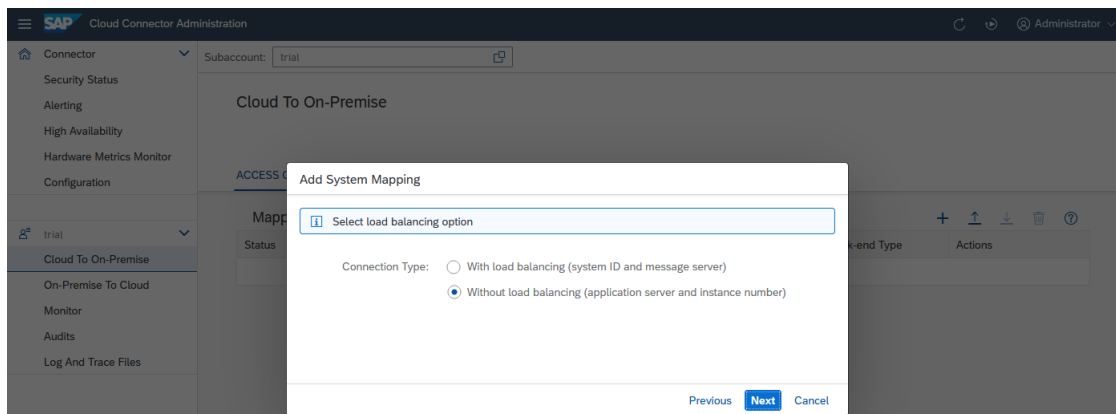
- Optional: In the Cloud Connector administration UI, you can check under *Audits* whether access has been denied:
Denying access for user DEMOUSER to system abapserver.hana.cloud:sapgw42 [connectionId=-1547299395]
- In the Cloud Connector administration UI, choose *Cloud To On-Premise* from your *Subaccount* menu, tab *Access Control*.
- In section *Mapping Virtual To Internal System* choose *Add* to define a new system.
 - For *Backend Type*, select **ABAP System** and choose *Next*.



- For *Protocol*, select **RFC** and choose *Next*.

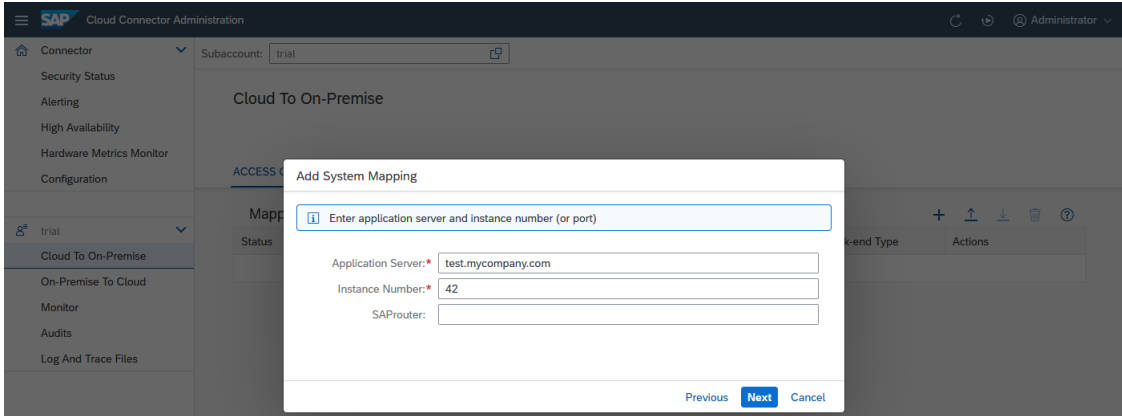


- Choose option *Without load balancing*.



4. Enter application server and instance number. The *Application Server* entry must be the physical host name of the machine on which the ABAP application server is running. Choose *Next*.

Example:

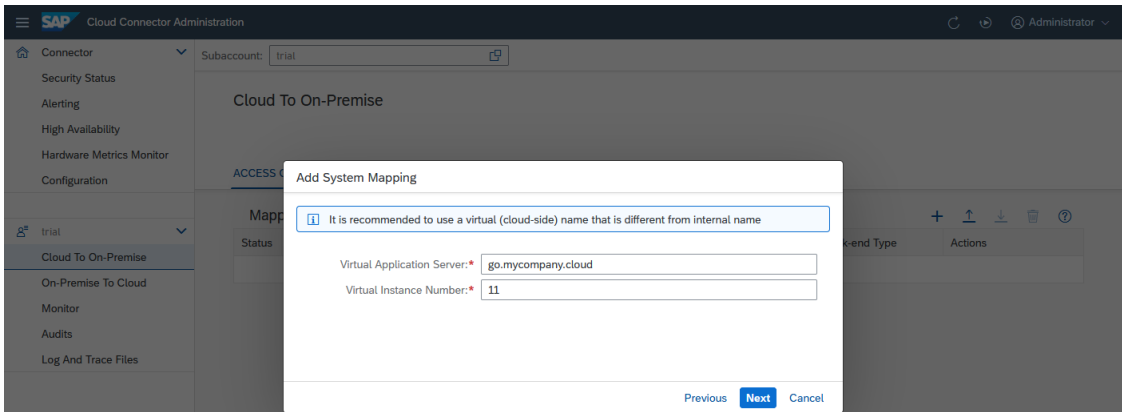


5. Enter server and instance number for virtual mapping.

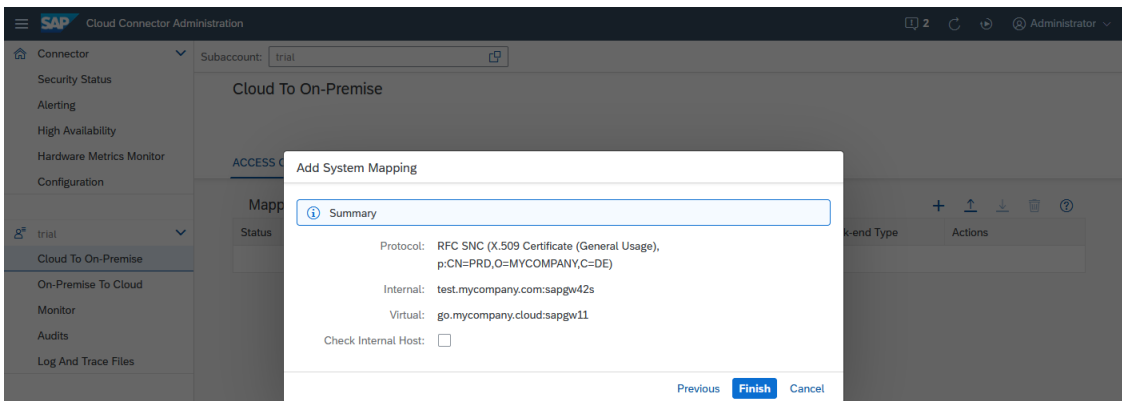
Note

The values must match with the ones of the destination configuration in the cloud cockpit.

Example:



6. Summary (example):



4. Call again the URL that references the cloud application in the Web browser. The application should now throw a different exception:

```

om.sap.conn.jco.JCoException: (102) JCO_ERROR_COMMUNICATION: Partner
signaled an error: Access denied for STFC_CONNECTION on
abapserver.hana.cloud:sapgw42. Expose the function module in your Cloud
Connector in case it was a valid request.
    at
com.sap.conn.jco.rt.MiddlewareJavaRfc.generateJCoException(MiddlewareJavaRfc.j
ava:632)
    at
com.sap.conn.jco.rt.MiddlewareJavaRfc$JavaRfcClient.execute(MiddlewareJavaRfc.
java:1764)
    at
com.sap.conn.jco.rt.ClientConnection.execute(ClientConnection.java:1110)
    at com.sap.conn.jco.rt.ClientConnection.execute(ClientConnection.java:943)
    at com.sap.conn.jco.rt.RfcDestination.execute(RfcDestination.java:1307)
    at com.sap.conn.jco.rt.RfcDestination.execute(RfcDestination.java:1278)
    at com.sap.conn.jco.rt.AbapFunction.execute(AbapFunction.java:295)
    at
com.sap.demo.jco.ConnectivityRFCExample.doGet(ConnectivityRFCExample.java:55)
    .... (cut rest of the call stack)

```

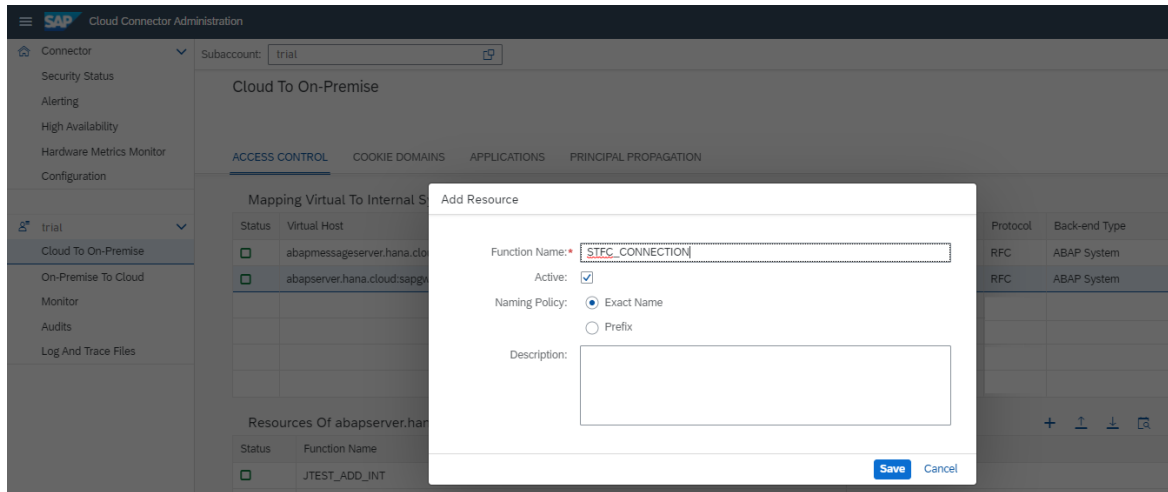
5. This means that the Cloud Connector denied invoking STFC_CONNECTION in this system. As a final step, you must provide access to this function module.

Back to [Steps \[page 300\]](#)

Configure the Function Module

The Cloud Connector only allows access to explicitly allowed resources (which, in an RFC scenario, are defined on the basis of function module names). To configure the function module, follow the steps below:

1. Optional: In the Cloud Connector administration UI, you can check under **Monitor** > **Audit** whether access has been denied:
Denying access for user DEMOUSER to resource STFC_CONNECTION on system abapserver.hana.cloud:sapgw42 [connectionId=609399452]
2. In the Cloud Connector administration UI, choose again *Cloud To On-Premise* from your *Subaccount* menu, and go to tab *Access Control*.
3. For the specified internal system referring to *abapserver.hana.cloud*, add a new resource. To do this, select the system in the table.
4. Add a new function name under the list of exposed resources. In section *Resources Accessible On abapserver.hana.cloud:sapgw42*, choose the *Add* button and specify STFC_CONNECTION as accessible resource, as shown in the screenshot below. Make sure that you have selected the *Exact Name* option to only expose this specific function module.



5. Call again the URL that references the cloud application in the Web browser. The application should now return a message showing the export parameters of the function module.

See also [Configure Access Control \(RFC\) \[page 465\]](#).

Back to [Steps \[page 300\]](#)

Next Step (Optional)

- [Monitoring Your Web Application \[page 304\]](#)

1.1.4.3.1.1.8 Monitoring Your Web Application

Monitor the state and logs of your Web application deployed on SAP BTP, using the Application Logging service.

For this purpose, create an instance of the Application Logging service (as you did for the Destination and Connectivity service) and bind it to your application, see [Create and Bind Service Instances \[page 289\]](#).

To activate JCo logging, set the following property in the *env* section of your manifest file:

```
SET_LOGGING_LEVEL: '{com.sap.conn.jco: INFO, com.sap.core.connectivity.jco: INFO}'
```

Now you can see and open the logs in the cloud cockpit or in the Kibana Dashboard in the tab [Logs](#), if you are within your application.

For error analysis, you can activate more detailed JCo debug tracing:

```
SET_LOGGING_LEVEL: '{com.sap.conn.jco: DEBUG, com.sap.core.connectivity.jco:
DEBUG}'
```

To include other relevant components for debug tracing, set:




```
SET_LOGGING_LEVEL: '{com.sap.conn.jco: DEBUG, com.sap.core.connectivity.jco:
DEBUG, com.sap.core.connectivity.apixt: DEBUG, com.sap.cloud.security: DEBUG,
com.sap.xs.security: DEBUG, com.sap.xs.env: DEBUG}'
```






1.1.4.3.1.2 Invoke ABAP Function Modules in Cloud ABAP Systems

Call a function module in a cloud ABAP system via RFC, using a sample Web application (Cloud Foundry environment).

This scenario performs a remote function call to invoke an ABAP function module, by using the Destination service in the Cloud Foundry environment, as well as a Cloud Foundry application router.

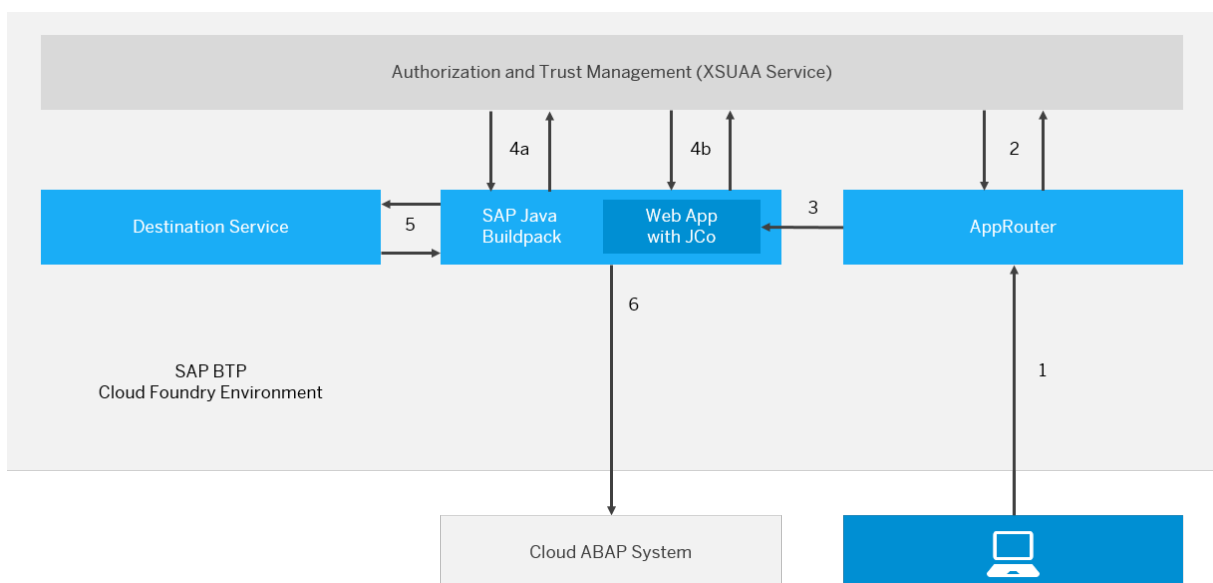
Tasks

Task Type	Task
 Operator and/or Developer	Scenario Overview [page 306]
	Used Values [page 307]
	Connectivity User Roles [page 307]
	Installation Prerequisites [page 308]
 Developer	Develop a Sample Web Application [page 308]
 Operator and/or Developer	Create and Bind Service Instances [page 313]

Task Type	Task
 Developer	Deploy the Application [page 315]
 Operator	Configure Roles and Trust [page 317]
 Operator and/or Developer	Set Up an Application Router [page 318]
 Operator	Configure the RFC Destination [page 322]
 Operator and/or Developer	Monitoring Your Web Application [page 323] (Optional)

Scenario Overview

Control Flow for Using the Java Connector (JCo) with Basic Authentication



Process Steps:

1. Call through AppRouter (entry point for business applications).

Note

AppRouter is only required you want to use multitenancy or perform user-specific service calls. In all other cases, JCo uses [cloud-security-xsuaa-integration](#) with `ClientCredentialFlow`.

2. Redirect to XSUAA for login. JSON Web Token (JWT1) is sent to AppRouter and cached there.
3. AppRouter calls Web app and sends JWT1 with credentials.
4. Buildpack/XSUAA interaction: Buildpack requests JWT2 to access the Destination service instance (JCo call).
5. Buildpack requests destination configuration (JWT2).
6. Buildpack sends request to the cloud ABAP system (with JWT2 and Authorization Header).

Since token exchanges are handled by the buildpack, you must only create and bind the service instances, see [Create and Bind Service Instances \[page 289\]](#).

Back to [Tasks \[page 305\]](#)

Used Values

This scenario uses:

- A subaccount in region **Europe (Frankfurt)**, for which the API endpoint is `api.cf.eu10.hana.ondemand.com`.
- The application name `jco-demo-<subaccount name>`, where `<subaccount name>` is the subdomain name of the subaccount. For this example, we use `p1234`.

Back to [Tasks \[page 305\]](#)

Connectivity User Roles

Different user roles are involved in the cloud-to-cloud connectivity scenario. The particular steps for the relevant roles are described below:

Application Developer

Develops Web applications using destinations. Scenario steps:

1. Installs Eclipse IDE, the Cloud Foundry Plugin for Eclipse and the Cloud Foundry CLI.
2. Develops a Java EE application using the JCo APIs.
3. Configures connectivity destinations via the SAP BTP cockpit.
4. Deploys and tests the Java EE application on SAP BTP.

Account Operator

Deploys Web applications, creates application routers, creates and binds service instances, conducts tests.
Scenario steps:

1. Obtains a ready Java EE application WAR file.
2. Deploys an application router with respective routes to the application.
3. Creates an XSUAA service instance and binds it to the router.
4. Deploys the Java EE application to an SAP BTP subaccount.
5. Creates a Destination service instance, and binds it to the application.
6. Creates and manages roles and role collections.

Back to [Tasks \[page 305\]](#)

Installation Prerequisites

- You have downloaded and set up your Eclipse IDE and the [Eclipse Tools for Cloud Foundry](#) .
- You have downloaded the Cloud Foundry CLI, see [Tools](#).

Back to [Tasks \[page 305\]](#)

Next Steps

- [Develop a Sample Web Application \[page 308\]](#)
- [Create and Bind Service Instances \[page 313\]](#)
- [Deploy the Application \[page 315\]](#)
- [Configure Roles and Trust \[page 317\]](#)
- [Set Up an Application Router \[page 318\]](#)
- [Configure the RFC Destination \[page 322\]](#)
- [Monitoring Your Web Application \[page 323\]](#) (Optional)

Related Information

[Multitenancy for JCo Applications \(Advanced\) \[page 323\]](#)

1.1.4.3.1.2.1 Develop a Sample Web Application


Create a Web application to call an ABAP function module via RFC.

Steps


1. [Create a Dynamic Web Project \[page 285\]](#)
2. [Include JCo Dependencies \[page 286\]](#)
3. [Create a Sample Servlet \[page 287\]](#)

Create a Dynamic Web Project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. On the *Project Explorer* view, choose **New** **Dynamic Web Project** in the context menu.
3. Enter `jco_demo` as the project name.
4. In the *Target Runtime* pane, select **Cloud Foundry**. If it is not yet in the list of available runtimes, choose *New Runtime* and select it from there.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish*.

 New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application. 

Project name:

Project location

Use default location

Location:

Target runtime

Dynamic web module version

Configuration

A good starting point for working with Cloud Foundry runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

Add project to an EAR

EAR project name:

Working sets

Add project to working sets

Working sets:

Back to [Steps \[page 284\]](#)

Include JCo Dependencies

To use JCo functionality seamlessly at compile time in Eclipse, you must include the JCo dependencies into your web project. Therefore, you must convert it into a maven project.

1. In the *Project Explorer* view, right-click on the project `jco-demo` and choose **Configure > Convert to Maven Project**.
2. In the dialog window, leave the default settings unchanged and choose *Finish*.
3. Open the `pom.xml` file and include the following dependency:

```
<dependencies>
  <dependency>
    <groupId>com.sap.cloud</groupId>
    <artifactId>neo-java-web-api</artifactId>
    <version>[3.71.8,4.0.0)</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

Back to [Steps \[page 284\]](#)

Create a Sample Servlet

1. From the `jco_demo` project node, choose **New > Servlet** in the context menu.
2. Enter `com.sap.demo.jco` as the `<package>` and `ConnectivityRFCExampleJava` as the `<Class name>`. Choose *Next*.
3. Choose *Finish* to create the servlet and open it in the Java editor.
4. Replace the entire servlet class to make use of the JCo API. The JCo API is visible by default for cloud applications. You do not need to add it explicitly to the application class path.

Sample Code

```
package com.sap.demo.jco;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.sap.conn.jco.AbapException;
import com.sap.conn.jco.JCoDestination;
import com.sap.conn.jco.JCoDestinationManager;
import com.sap.conn.jco.JCoException;
import com.sap.conn.jco.JCoFunction;
import com.sap.conn.jco.JCoParameterList;
import com.sap.conn.jco.JCoRepository;

/**
 * Sample application that makes use of the capability to invoke a
 * function module in an ABAP system
 * via RFC
 *
 * Note: The JCo APIs are available under com.sap.conn.jco.
 */
```

```

*/
@WebServlet("/ConnectivityRFCEExample/*")
public class ConnectivityRFCEExample extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        PrintWriter responseWriter = response.getWriter();
        try {
            // access the RFC Destination "JCoDemoSystem"
            JCoDestination destination =
JCoDestinationManager.getDestination("JCoDemoSystem");

            // make an invocation of STFC_CONNECTION in the backend;
            JCoRepository repo = destination.getRepository();
            JCoFunction stfcConnection =
repo.getFunction("STFC_CONNECTION");

            JCoParameterList imports =
stfcConnection.getImportParameterList();
            imports.setValue("REQUTEXT", "SAP BTP Connectivity runs with
JCo");

            stfcConnection.execute(destination);
            JCoParameterList exports =
stfcConnection.getExportParameterList();
            String echotext = exports.getString("ECHOTEXT");
            String resptext = exports.getString("RESPTTEXT");
            response.setHeader("Content-type", "text/html");
            responseWriter.println("<html><body>");
            responseWriter.println("<h1>Executed STFC_CONNECTION in system
JCoDemoSystem</h1>");
            responseWriter.println("<p>Export parameter ECHOTEXT of
STFC_CONNECTION:<br>");
            responseWriter.println(echotext);
            responseWriter.println("<p>Export parameter RESPTTEXT of
STFC_CONNECTION:<br>");
            responseWriter.println(resptext);
            responseWriter.println("</body></html>");
        } catch (AbapException ae) {
            // just for completeness: As this function module does not
have an exception
            // in its signature, this exception cannot occur. But you
should always
            // take care of AbapExceptions
        } catch (JCoException e) {
            response.setHeader("Content-type", "text/html");
            responseWriter.println("<html><body>");
            responseWriter
                .println("<h1>Exception occurred while executing
STFC_CONNECTION in system JCoDemoSystem</h1>");
            responseWriter.println("<pre>");
            e.printStackTrace(responseWriter);
            responseWriter.println("</pre>");
            responseWriter.println("</body></html>");
        }
    }
}

```

5. Save the Java editor and make sure that the project compiles without errors.

Back to [Steps \[page 284\]](#)

Next Steps

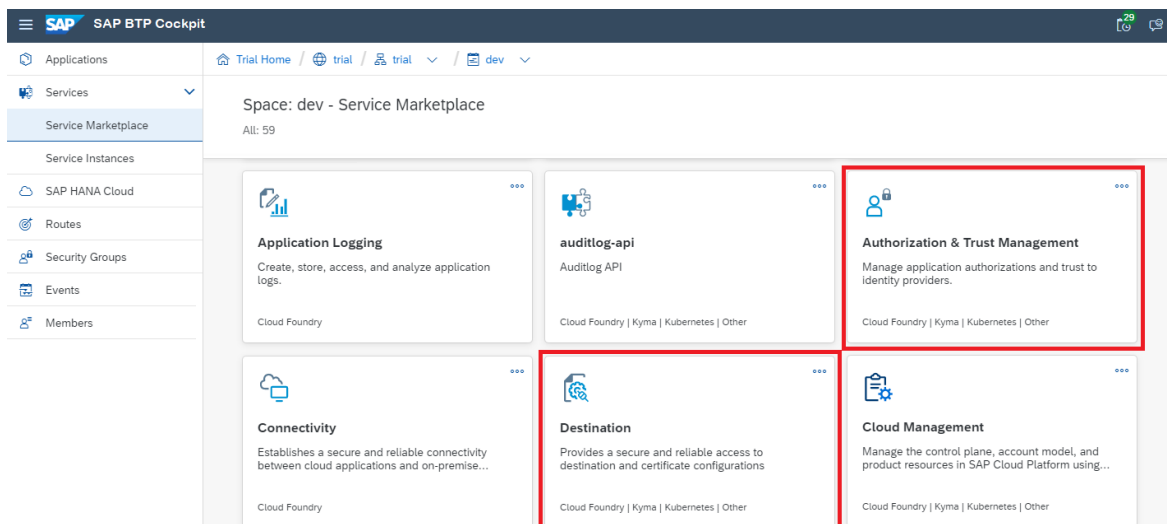
- [Create and Bind Service Instances](#) [page 313]
- [Deploy the Application](#) [page 315]
- [Configure Roles and Trust](#) [page 317]
- [Set Up an Application Router](#) [page 318]
- [Configure the RFC Destination](#) [page 322]
- [Monitoring Your Web Application](#) [page 323] (Optional)

1.1.4.3.1.2.2 Create and Bind Service Instances

You must create and bind several service instances, before you can use your application.

Procedure

1. Logon to the cloud cockpit and choose your subaccount.
2. Choose the space where you want to deploy your demo application.
3. Choose *Service Marketplace* and find these 2 services:
 - *Destination*
 - *Authorization & Trust Management* (XSUAA)



4. Create and bind a service instance for each of these services.
 - [Destination service](#) [page 290]
 - [Authorization & Trust Management \(XSUAA service\)](#) [page 291]

Destination Service

1. Choose ► *Destination* ► *Create Instance* ►.
2. Insert an instance name (for example, `destination_jco`) and choose *Create Instance*.

Back to [Procedure \[page 289\]](#)

Authorization & Trust Management (XSUAA Service)

1. Choose ► *Authorization & Trust Management* ► *Create Instance* ►
2. Select *<Service Plan>* **application**.
3. Enter an *<Instance Name>* and choose *Next*.

Note

The instance name must match the one defined in the manifest file.

4. In the next tab *Parameters*, insert the following as a JSON file:

Sample Code

```
{
  "xsappname" : "jco-demo-p1234",
  "tenant-mode": "dedicated",
  "scopes": [
    {
      "name": "$XSAPPNAME.all",
      "description": "all"
    }
  ],
  "role-templates": [
    {
      "name": "all",
      "description": "all",
      "scope-references": [
        "$XSAPPNAME.all"
      ]
    }
  ]
}
```

5. Go to tab *Review* and choose *Create Instance*.

Back to [Procedure \[page 289\]](#)

Next Steps

- [Deploy the Application \[page 315\]](#)
- [Configure Roles and Trust \[page 317\]](#)
- [Set Up an Application Router \[page 318\]](#)
- [Configure the RFC Destination \[page 322\]](#)
- [Monitoring Your Web Application \[page 323\]](#) (Optional)

1.1.4.3.1.2.3 Deploy the Application

Deploy your Cloud Foundry application to call an ABAP function module via RFC.

Prerequisites

You have created and bound the required service instances, see [Create and Bind Service Instances \[page 289\]](#).

Procedure

1. To deploy your Web application, you can use the following two alternative procedures:
 - [Deploying from the Eclipse IDE](#)
 - Deploying from the CLI, see [Developing Java in the Cloud Foundry Environment](#)
2. In the following, we publish it with the CLI.
3. To do this, create a `manifest.yml` file. The key parameter is `USE_JCO: true`, which must be set to include JCo into the buildpack during deployment.

Note

JCo supports the usage of X.509 secrets for communication with the Destination/Connectivity service. If you want to use it, you must specify the binding accordingly.

For more information, see [Binding Parameters of SAP Authorization and Trust Management Service](#).

manifest.yml

Sample Code

```
---
applications:
- name: jco-demo-p1234
  buildpacks:
```

```

- sap_java_buildpack
env:
  USE_JCO: true
  # This is necessary only if more than one instance is bound
  xsuaa_connectivity_instance_name: "xsuaa_jco"
  connectivity_instance_name: "connectivity_jco"
  destination_instance_name: "destination_jco"
services:
  - xsuaa_jco
  - connectivity_jco
  - destination_jco

```

⚠ Caution

The client libraries (java-security, spring-xsuaa, and container security api for node.js as of version 3.0.6) have been updated. When using these libraries, setting the parameter `SAP_JWT_TRUST_ACL` has become obsolete. This update comes with a change regarding scopes:

- For a business application A calling an application B, it is now mandatory that application B grants at least one scope to the calling business application A.
- Business application A must accept these granted scopes or authorities as part of the application security descriptor.

You can grant scopes using the `xs-security.json` file.

For more information, see [Application Security Descriptor Configuration Syntax](#), specifically the sections *Referencing the Application* and *Authorities*.

📄 Note

If you have more than one instance of those three services bound to your application, you must specify which one JCo should use with the respective env parameters:

- `xsuaa_connectivity_instance_name`
- `connectivity_instance_name`
- `destination_instance_name`.

4. In Eclipse, right-click on the project and navigate to **Export > WAR file**.
5. Choose a destination by pressing the *Browse...* button next to the `manifest.yml` you created before, for example as `jcodemo.war`.
6. Leave the other default settings unchanged and choose *Finish* to export the WAR file.
7. Perform a CLI login via `cf login -a api.cf.eu10.hana.ondemand.com -u <your_email_address>` (password, org and space are prompted after a successful login).
8. Push the application with `cf push -f manifest.yml -p jcodemo.war`.
9. Now, the application should be deployed successfully.

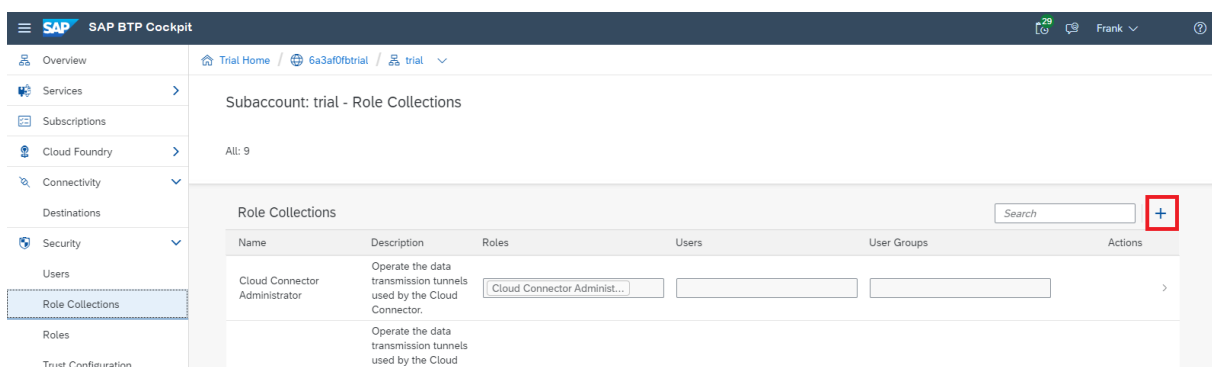
Next Steps

- [Configure Roles and Trust \[page 317\]](#)
- [Set Up an Application Router \[page 318\]](#)
- [Configure the RFC Destination \[page 322\]](#)
- [Monitoring Your Web Application \[page 323\]](#) (Optional)

1.1.4.3.1.2.4 Configure Roles and Trust

Configure a role that enables your user to access your Web application.

To add and assign roles, navigate to the subaccount view of the cloud cockpit and choose **Security > Role Collections**.



1. Create a new role collection with the name **a11**.
2. From the subaccount menu, choose *Trust Configuration*.
3. If you don't have a trust configuration, follow the steps in [Manually Establish Trust and Federation Between SAP Authorization and Trust Management Service and Identity Authentication](#).
4. Click on the IdP name of your choice.
5. Type in your e-mail address and choose *Show Assignments*.
6. If your user has not yet been added to the SAP ID service, you see following popup. In this case, add your user now.

Confirmation

To see and assign role collections, you must first add `<user>@sap.com` as a user of identity provider SAP ID Service.

[Add User](#) [Cancel](#)

7. You should now be able to click *Assign Role Collection*. Choose role collection **a11** and assign it.

Next Steps

- [Set Up an Application Router \[page 318\]](#)

- [Configure the RFC Destination \[page 322\]](#)
- [Monitoring Your Web Application \[page 323\]](#) (Optional)

Related Information

[Working with Role Collections](#)

1.1.4.3.1.2.5 Set Up an Application Router

For authentication purposes, configure and deploy an application router for your test application.

Note

AppRouter is only required if you want to use multitenancy or perform user-specific service calls. In all other cases, JCo uses [cloud-security-xsuaa-integration](#) with `ClientCredentialFlow`.

1. To set up an application router, follow the steps in [Application Router](#) or use the demo file [approuter.zip](#) ([download](#)).
2. For deployment, you need a manifest file, similar to this one:

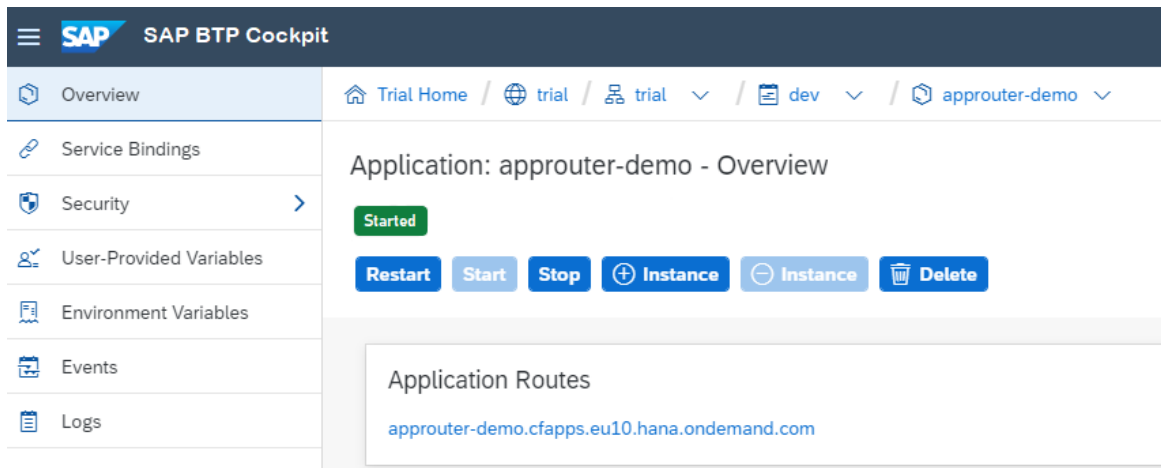
Sample Code

```
---
applications:
- name: approuter-jco-demo-p1234
  path: ./
  buildpacks:
  - nodejs_buildpack
  memory: 120M
  routes:
  - route: approuter-jco-demo-p1234.cfapps.eu10.hana.ondemand.com
  env:
    NODE_TLS_REJECT_UNAUTHORIZED: 0
    destinations: >
      [
        { "name": "dest-to-example", "url" : "https://jco-
demo-p1234.cfapps.eu10.hana.ondemand.com/ConnectivityRFCExample",
"forwardAuthToken": true }
      ]
  services:
  - xsuaa_jco
```

Note

- The routes and destination URLs need to fit your test application.
- In this example, we already bound our XSUAA instance to the application router. Alternatively, you could also do this via the cloud cockpit.

3. Push the approuter with `cf push -f manifest.yml -p approuter.zip`.
4. To navigate to the `approuter` application in the cloud cockpit, choose `>> <your_space> > Applications > <your application> > Overview >`.



5. When choosing the application route, you are requested to login. Provide the credentials known by the IdP you configured in *Roles & Trust*.
6. After successful login, you are routed to the test application which is then executed.
7. If the application issues an exception, saying that the `JCoDemoSystem` destination has not yet been specified, you must configure the `JCoDemoSystem` destination first.

```
Exception occurred while executing STFC_CONNECTION in system JCoDemoSystem
com.sap.conn.jco.JCoException: (106) JCO_ERROR_RESOURCE: Destination
JCoDemoSystem does not exist
    at
    com.sap.conn.jco.rt.DefaultDestinationManager.update(DefaultDestinationManager
.java:223)
    at
    com.sap.conn.jco.rt.DefaultDestinationManager.searchDestination(DefaultDestina
tionManager.java:377)
    at
    com.sap.conn.jco.rt.DefaultDestinationManager.getDestinationInstance(DefaultDe
stinationManager.java:96)
    at
    com.sap.conn.jco.JCoDestinationManager.getDestination(JCoDestinationManager.ja
va:52)
    at
    com.sap.demo.jco.ConnectivityRFCExample.doGet(ConnectivityRFCExample.java:47)
    .... (cut rest of the call stack)
```

Note

Make sure you **don't** include this dependency

```
<dependency>
  <groupId>com.sap.cloud.security</groupId>
  <artifactId>java-security</artifactId>
</dependency>
```

or any of its dependencies such as `java-api` with scope `compile` directly or transitively with any other jar.

Calling JCo APIs from Newly Created Threads

If you are using an Application Router and it is mandatory for you to call JCo APIs from a different thread than the one which is executing your servlet function, make sure the thread local information of the [cloud-security-xsuaa-integration API](#), used by JCo internally, is set again within your newly created thread.

To do this, add the following dependency to your project:

```
<dependency>
  <groupId>com.sap.cloud.security</groupId>
  <artifactId>java-api</artifactId>
  <version>2.7.7</version>
  <scope>provided</scope>
</dependency>
```

Adjust your code from the step [Develop a Sample Web Application \[page 308\]](#) in the following way:

Sample Code

```
package com.sap.demo.jco;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.sap.cloud.security.token.SecurityContext;
import com.sap.cloud.security.token.Token;
import com.sap.conn.jco.AbapException;
import com.sap.conn.jco.JCoDestination;
import com.sap.conn.jco.JCoDestinationManager;
import com.sap.conn.jco.JCoException;
import com.sap.conn.jco.JCoFunction;
import com.sap.conn.jco.JCoParameterList;
import com.sap.conn.jco.JCoRepository;
/**
 *
 * Sample application that uses the connectivity service. In particular, it is
 * making use of the capability to invoke a function module in an ABAP system
 * via RFC
 *
 * Note: The JCo APIs are available under <code>com.sap.conn.jco</code>.
 */
@WebServlet("/ConnectivityRFCEXample/*")
public class ConnectivityRFCEXample extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        PrintWriter responseWriter = response.getWriter();
        // access the token from the thread which is executing the servlet
        Token token = SecurityContext.getToken();
        Thread runThread = new Thread(() -> {
            // set the information in the newly created thread
            SecurityContext.setToken(token);
            try {
                // access the RFC Destination "JCoDemoSystem"
                JCoDestination destination =
JCoDestinationManager.getDestination("JCoDemoSystem_Normal");
                // make an invocation of STFC_CONNECTION in the backend
```


```

        JCoRepository repo = destination.getRepository();
        JCoFunction stfcConnection =
repo.getFunction("STFC_CONNECTION");
        JCoParameterList imports =
stfcConnection.getImportParameterList();
        imports.setValue("REQUTEXT", "SAP BTP Connectivity runs with
JCo");

        stfcConnection.execute(destination);
        JCoParameterList exports =
stfcConnection.getExportParameterList();
        String echotext = exports.getString("ECHOTEXT");
        String resptext = exports.getString("RESPTEXT");
        response.addHeader("Content-type", "text/html");
        responseWriter.println("<html><body>");
        responseWriter.println("<h1>Executed STFC_CONNECTION in
system JCoDemoSystem</h1>");
        responseWriter.println("<p>Export parameter ECHOTEXT of
STFC_CONNECTION:<br>");
        responseWriter.println(echotext);
        responseWriter.println("<p>Export parameter RESPTEXT of
STFC_CONNECTION:<br>");
        responseWriter.println(resptext);
        responseWriter.println("</body></html>");
    } catch (AbapException ae) {
        // just for completeness: As this function module does not
have an exception
        // in its signature, this exception cannot occur. But you
should always
        // take care of AbapExceptions
    } catch (JCoException e) {
        response.addHeader("Content-type", "text/html");
        responseWriter.println("<html><body>");
        responseWriter
            .println("<h1>Exception occurred while executing
STFC_CONNECTION in system JCoDemoSystem</h1>");
        responseWriter.println("<pre>");
        e.printStackTrace(responseWriter);
        responseWriter.println("</pre>");
        responseWriter.println("</body></html>");
    } finally {
        // after execution clear the token again
        SecurityContext.clearToken();
    }
    });
    runThread.start();
    // wait to be finished
    try {
        runThread.join();
    } catch (InterruptedException e) {
        e.printStackTrace(responseWriter);
    }
}
}
}

```

📌 Note

If you want to use [thread pools](#) , make sure that in your thread pool implementation this information is set correctly in the thread which is about to be (re)used, and removed as soon as the thread is put back into the pool.

Next Steps

- [Configure the RFC Destination \[page 322\]](#)
- [Monitoring Your Web Application \[page 323\]](#) (Optional)

1.1.4.3.1.2.6 Configure the RFC Destination

Configure an RFC destination on SAP BTP that you can use in your Web application to call the cloud ABAP system.

To configure the destination, you must use a WebSocket application server host name (`<id>.abap.eu10.hana.ondemand.com`) and a WebSocket port (`443`).

1. Create a `.properties` file with the following settings:

```
Name=JCoDemoSystem
Type=RFC
jco.client.wshost= <id>.abap.eu10.hana.ondemand.com
jco.client.wsport=443
jco.client.alias_user=<DEMOUSER>
jco.client.passwd=<Password>
jco.client.client=100
jco.client.lang=EN
jco.destination.pool_capacity=5
jco.destination.proxy_type=Internet
```

2. Go to your subaccount in the cloud cockpit.
 1. From the subaccount menu, choose **Connectivity** > **Destinations** > **Import Destination** and upload this file.
 2. Alternatively, you can create a destination for the service instance `destination_jco` to make it visible only for this instance. To do this, go to **<your space>** > **Services** > **Service Instances** > **<your destination_instance>** and choose **Destinations**.
3. Specify a trust/key store or security information, see [WebSocket Connection \[page 165\]](#).

Next Steps

- [Monitoring Your Web Application \[page 323\]](#) (Optional)

Related Information

[Target System Configuration \[page 163\]](#)

1.1.4.3.1.2.7 Monitoring Your Web Application

Monitor the state and logs of your Web application deployed on SAP BTP, using the Application Logging service.

For this purpose, create an instance of the Application Logging service (as you did for the Destination service) and bind it to your application, see [Create and Bind Service Instances \[page 289\]](#).

To activate JCo logging, set the following property in the `env` section of your manifest file:

```
SET_LOGGING_LEVEL: '{com.sap.conn.jco: INFO, com.sap.core.connectivity.jco: INFO}'
```

Now you can see and open the logs in the cloud cockpit or in the Kibana Dashboard in the tab [Logs](#), if you are within your application.

For error analysis, you can activate more detailed JCo debug tracing:

```
SET_LOGGING_LEVEL: '{com.sap.conn.jco: DEBUG, com.sap.core.connectivity.jco: DEBUG}'
```

To include other relevant components for debug tracing, set:

```
SET_LOGGING_LEVEL: '{com.sap.conn.jco: DEBUG, com.sap.core.connectivity.jco: DEBUG, com.sap.core.connectivity.apixt: DEBUG, com.sap.cloud.security: DEBUG, com.sap.xs.security: DEBUG, com.sap.xs.env: DEBUG}'
```

1.1.4.3.1.3 Multitenancy for JCo Applications (Advanced)

Learn about the required steps to make your Cloud Foundry JCo application tenant-aware.

Using this procedure, you can enable the sample JCo application created in [Invoke ABAP Function Modules in On-Premise ABAP Systems \[page 280\]](#) or [Invoke ABAP Function Modules in Cloud ABAP Systems \[page 305\]](#), for multitenancy.

Steps

1. [Prerequisites \[page 324\]](#)
2. [Adjust the Application Router \[page 324\]](#)
3. [Adjust the XSUAA Service Instance and Roles \[page 325\]](#)
4. [Make the Application Subscribable \[page 326\]](#)
5. [Create the SAAS Provisioning Service \[page 331\]](#)

6. [Subscribe to the Application \[page 332\]](#)
7. [Create a New Route \[page 334\]](#)

Prerequisites

- Your runtime environment uses SAP Java Buildpack version 1.9.0 or higher.
- You have successfully completed one of these precedures:
[Invoke ABAP Function Modules in On-Premise ABAP Systems \[page 280\]](#)
[Invoke ABAP Function Modules in Cloud ABAP Systems \[page 305\]](#)
- You have created a second subaccount (in the same global account), that is used to subscribe to your application.

Back to [Steps \[page 323\]](#)

Adjust the Application Router

The application router needs to be tenant-aware with a `TENANT_HOST_PATTERN` to recognize different tenants from the URL, see [Multitenancy](#). `TENANT_HOST_PATTERN` should have the following format: `^(.*)<application domain>`. The application router extracts the token captured by `(.*)` to use it as the subscriber tenant. The manifest file might look like this:

Sample Code

```
manifest.yml
---
applications:

- name: approuter-jco-demo-p42424242
  path: ./
  buildpacks:
  - nodejs_buildpack
  memory: 120M
  routes:
  - route: approuter-jco-demo-p42424242.cfapps.eu10.hana.ondemand.com
  env:
    TENANT_HOST_PATTERN: "^(.*)approuter-jco-demo-
p42424242.cfapps.eu10.hana.ondemand.com"
    NODE_TLS_REJECT_UNAUTHORIZED: 0
  destinations: >
    [
      { "name": "dest-to-example", "url" : "https://jco-
demo-p42424242.cfapps.eu10.hana.ondemand.com/ConnectivityRFCEExample",
"forwardAuthToken": true }
    ]
  services:
  - xsuaa_jco
```

Back to [Steps \[page 323\]](#)

Adjust the XSUAA Service Instance and Roles

To call the XSUAA in a tenant-aware way, you must adjust the configuration JSON file. The tenant mode must now have the value "shared". Also, you must allow calling the previously defined REST APIs (callbacks).

Sample Code

```
{
  "xsappname" : "jco-demo-p42424242",
  "tenant-mode": "shared",
  "scopes": [
    {
      "name": "$XSAPPNAME.Callback",
      "description": "With this scope set, the callbacks for tenant
onboarding, offboarding and getDependencies can be called.",
      "grant-as-authority-to-apps": [
        "$XSAPPNAME(application,sap-provisioning,tenant-onboarding)"
      ]
    },
    {
      "name": "$XSAPPNAME.access",
      "description": "app access"
    },
    {
      "name": "uaa.user",
      "description": "uaa.user"
    }
  ],
  "role-templates": [
    {
      "name": "MultitenancyCallbackRoleTemplate",
      "description": "Call callback-services of applications",
      "scope-references": [
        "$XSAPPNAME.Callback"
      ]
    },
    {
      "name": "UAAaccess",
      "description": "UAA user access",
      "scope-references": [
        "uaa.user",
        "$XSAPPNAME.access"
      ]
    }
  ]
}
```

Add Roles

1. In the cloud cockpit, navigate to the subaccount view and go the tab *Role Collections* under *Security* (see step [Configure Roles and Trust \[page 294\]](#) from the previous procedure).
2. Click on the role collection name.
3. Choose *Add Role*.
4. In the popup window, select the demo application as *<Application Identifier>*.

5. For `<Role Template>` and `<Role>`, use **MultitenancyCallbackRoleTemplate** and choose *Save*.
6. Choose *Add Role* again.
7. Select the demo application as `<Application Identifier>`.
8. For *Role Template* and *Role* use **UAAAccess** and choose *Save*.

Back to [Steps \[page 323\]](#)

Make the Application Subscribable

Firstly, in order to make the application subscribable, it must provide at least the following REST APIs:

- [GET dependent services of an application \[page 326\]](#)
- [PUT tenant subscription to an application \[page 329\]](#)

In our sample application, we implement new servlets for each of these APIs.

The following servlets need additional maven dependencies:

Sample Code

```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.8.5</version>
</dependency>
<dependency>
  <groupId>com.unboundid.components</groupId>
  <artifactId>json</artifactId>
  <version>1.0.0</version>
</dependency>
<dependency>
  <groupId>javax.ws.rs</groupId>
  <artifactId>javax.ws.rs-api</artifactId>
  <version>2.1.1</version>
</dependency>
```

GET Dependencies

The current JCo dependencies are the Connectivity and Destination service. Thus, the GET API must return information about these two services:

Sample Code

```
import java.io.IOException;
import java.util.Arrays;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```

import javax.ws.rs.core.MediaType;

import org.json.JSONException;
import org.json.JSONObject;

import com.google.gson.Gson;

@WebServlet("/callback/v1.0/dependencies")
public class GetDependencyServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private static final String DESTINATION_SERVICE_NAME = "destination";
    private static final String CONNECTIVITY_SERVICE_NAME = "connectivity";
    private static final String XSAPPNAME_PROPERTY = "xsappname";

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        try {
            DependantServiceDto destinationService =
createLPSDependency(DESTINATION_SERVICE_NAME);
            DependantServiceDto connectivityService =
createLPSDependency(CONNECTIVITY_SERVICE_NAME);
            List<DependantServiceDto> dependenciesList =
Arrays.asList(destinationService, connectivityService);

            response.setStatus(200);
            response.setContentType(MediaType.APPLICATION_JSON);

            String json = new Gson().toJson(dependenciesList);
            response.getWriter().println(json);
        } catch (JSONException e) {
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
e.getMessage());
        }
    }

    private static DependantServiceDto createLPSDependency(String
serviceName) throws JSONException {
        JSONObject credentials =
EnvironmentVariableAccessor.getServiceCredentials(serviceName);
        String xsappname = credentials.getString(XSAPPNAME_PROPERTY);
        return new DependantServiceDto(serviceName, xsappname);
    }
}

```

Find the code of the two helper classes below:

DependantServiceDto.java

```

public class DependantServiceDto {
    private String appName;
    private String appId;
    public DependantServiceDto() {}
    public DependantServiceDto(String appName, String appId) {
        this.appName = appName;
        this.appId = appId;
    }
    public String getAppName() {
        return appName;
    }
    public void setAppName(String appName) {
        this.appName = appName;
    }
    public String getAppId() {

```

```

        return appId;
    }
    public void setAppId(String appId) {
        this.appId = appId;
    }
    @Override public boolean equals(Object o) {
        if (this == o)
            return true;
        if (!(o instanceof DependantServiceDto))
            return false;
        DependantServiceDto that = (DependantServiceDto) o;
        if (!appName.equals(that.appName))
            return false;
        return appId.equals(that.appId);
    }
    @Override public int hashCode() {
        int result = appName.hashCode();
        result = 31 * result + appId.hashCode();
        return result;
    }
}

```

EnvironmentVariableAccessor.java

```

import java.text.MessageFormat;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
/**
 * Methods for extracting configurations from the environment variables
 */
public final class EnvironmentVariableAccessor
{
    public static final String BEARER_WITH_TRAILING_SPACE="Bearer ";
    private static final String VCAP_SERVICES=System.getenv("VCAP_SERVICES");
    private static final String VCAP_SERVICES_CREDENTIALS="credentials";
    private static final String VCAP_SERVICES_NAME="name";
    private static final String
PROP_XSUAA_CONNECTIVITY_INSTANCE_NAME="xsuaa_connectivity_instance_name";
    private static final String DEFAULT_XSUAA_CONNECTIVITY_INSTANCE_NAME="conn-
xsuaa";
    private EnvironmentVariableAccessor()
    {
    }
    /**
     * Returns service credentials for a given service from VCAP_SERVICES
     *
     * @see <a href= "https://docs.run.pivotal.io/devguide/deploy-apps/
environment-variable.html#VCAP-SERVICES">VCAP_SERVICES</a>
     */
    public static JSONObject getServiceCredentials(String serviceName) throws
JSONException
    {
        return new
JSONObject(VCAP_SERVICES).getJSONArray(serviceName).getJSONObject(0).getJSONObjec
t(VCAP_SERVICES_CREDENTIALS);
    }
    /**
     * Returns service credentials for a given service instance from
VCAP_SERVICES
     *
     * @see <a href= "https://docs.run.pivotal.io/devguide/deploy-apps/
environment-variable.html#VCAP-SERVICES">VCAP_SERVICES</a>
     */
    public static JSONObject getServiceCredentials(String serviceName, String
serviceInstanceName) throws JSONException

```

```

    {
        JSONArray jsonarr=new
JSONObject(VCAP_SERVICES).getJSONArray(serviceName);
        for (int i=0; i<jsonarr.length(); i++)
        {
            JSONObject serviceInstanceObject=jsonarr.getJSONObject(i);
            String
instanceName=serviceInstanceObject.getString(VCAP_SERVICES_NAME);
            if (instanceName.equals(serviceInstanceName))
            {
                return
serviceInstanceObject.getJSONObject(VCAP_SERVICES_CREDENTIALS);
            }
        }
        throw new RuntimeException(MessageFormat.format("Service instance {0} of
service {1} not bound to application", serviceInstanceName, serviceName));
    }
    /**
     * Returns service credentials attribute for a given service from
VCAP_SERVICES
     *
     * @see <a href= "https://docs.run.pivotal.io/devguide/deploy-apps/
environment-variable.html#VCAP-SERVICES">VCAP_SERVICES</a>
     */
    public static String getServiceCredentialsAttribute(String serviceName,
String attributeName) throws JSONException
    {
        return getServiceCredentials(serviceName).getString(attributeName);
    }
    /**
     * Returns the name of the xsuaa service for connectivity service.
     */
    public static String getXsuaaConnectivityInstanceName()
    {
        String
xsuaaConnectivityInstanceName=System.getenv(PROP_XSUAA_CONNECTIVITY_INSTANCE_NAME
);
        return xsuaaConnectivityInstanceName!=null?
xsuaaConnectivityInstanceName:DEFAULT_XSUAA_CONNECTIVITY_INSTANCE_NAME;
    }
}

```

Back to [Make the Application Subscribable \[page 326\]](#)

PUT Tenant Subscription

This API is called whenever a tenant is subscribing. In our example, we just read the received JSON, and return the tenant-aware URL of the application router which points to our application. Also, if a tenant wants to unsubscribe, DELETE does currently nothing.

Sample Code

```

SubscribeServlet
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;

```



```

@WebServlet("/callback/v1.0/tenants/*")
public class SubscribeServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPut(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        PayloadDataDto payload = new Gson().fromJson(request.getReader(),
PayloadDataDto.class);
        response.getWriter().println("https://" +
payload.getSubscribedSubdomain() + ".approuter-jco-
demo.cfapps.eu10.hana.ondemand.com");
    }

    @Override
    protected void doDelete(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        super.doDelete(req, resp);
    }
}

```

Here is the helper class:

PayloadDataDto.java

```

import java.util.Map;
public class PayloadDataDto {
    private String subscriptionAppName;
    private String subscriptionAppId;
    private String subscribedTenantId;
    private String subscribedSubdomain;
    private String subscriptionAppPlan;
    private long subscriptionAppAmount;
    private String[] dependantServiceInstanceAppIds = null;
    private Map<String, String> additionalInformation;
    public PayloadDataDto() {}
    public PayloadDataDto(String subscriptionAppName, String subscriptionAppId,
String subscribedTenantId, String subscribedSubdomain, String
subscriptionAppPlan,
Map<String, String> additionalInformation) {
        this.subscriptionAppName = subscriptionAppName;
        this.subscriptionAppId = subscriptionAppId;
        this.subscribedTenantId = subscribedTenantId;
        this.subscribedSubdomain = subscribedSubdomain;
        this.subscriptionAppPlan = subscriptionAppPlan;
        this.additionalInformation = additionalInformation;
    }
    public String getSubscriptionAppName() {
        return subscriptionAppName;
    }
    public void setSubscriptionAppName(String subscriptionAppName) {
        this.subscriptionAppName = subscriptionAppName;
    }
    public String getSubscriptionAppId() {
        return subscriptionAppId;
    }
    public void setSubscriptionAppId(String subscriptionAppId) {
        this.subscriptionAppId = subscriptionAppId;
    }
    public String getSubscribedTenantId() {
        return subscribedTenantId;
    }
    public void setSubscribedTenantId(String subscribedTenantId) {
        this.subscribedTenantId = subscribedTenantId;
    }
}

```

```

    }
    public String getSubscribedSubdomain() {
        return subscribedSubdomain;
    }
    public void setSubscribedSubdomain(String subscribedSubdomain) {
        this.subscribedSubdomain = subscribedSubdomain;
    }
    public String getSubscriptionAppPlan() {
        return subscriptionAppPlan;
    }
    public void setSubscriptionAppPlan(String subscriptionAppPlan) {
        this.subscriptionAppPlan = subscriptionAppPlan;
    }
    public Map<String, String> getAdditionalInformation() {
        return additionalInformation;
    }
    public void setAdditionalInformation(Map<String, String>
additionalInformation) {
        this.additionalInformation = additionalInformation;
    }
    public long getSubscriptionAppAmount() {
        return subscriptionAppAmount;
    }
    public void setSubscriptionAppAmount(long subscriptionAppAmount) {
        this.subscriptionAppAmount = subscriptionAppAmount;
    }
    public String[] getDependantServiceInstanceAppIds() {
        return dependantServiceInstanceAppIds;
    }
    public void setDependantServiceInstanceAppIds(
        String[] dependantServiceInstanceAppIds) {
        this.dependantServiceInstanceAppIds = dependantServiceInstanceAppIds;
    }
    public String toString() {
        return String.format("Payload data: subscriptionAppName=%s,
subscriptionAppId=%s, subscribedTenantId=%s,"
            + "subscribedSubdomain=%s subscriptionAppPlan=%s
subscriptionAppAmount=%s dependantServiceInstanceAppIds=%s",
            this.subscriptionAppName, this.subscriptionAppId,
                this.subscribedTenantId, this.subscribedSubdomain,
                this.subscriptionAppPlan, this.getSubscriptionAppAmount(),
                dependantServiceInstanceAppIds);
    }
}

```

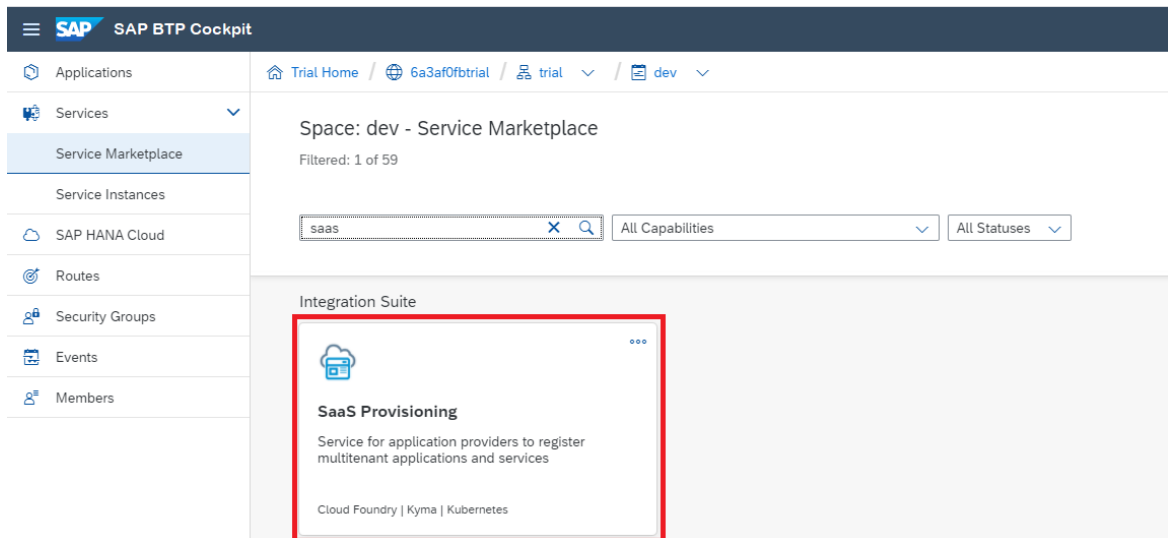
Back to [Make the Application Subscribable \[page 326\]](#)

Back to [Steps \[page 323\]](#)

Create the SAAS Provisioning Service

For the subscription of other tenants, your application must have a bound SAAS provisioning service instance. You can do this using the cockpit:

1. Go to the [Service Marketplace](#) in the cloud cockpit:



2. Choose **SaaS Provisioning** **Instances** **New Instance**.
3. Select **application** as **<Service Plan>** and choose **Next**.
4. In the step **Specify Parameters (Optional)**, insert the following as a JSON file:

```

Sample Code

{
  "xsappname" : "jco-demo-p42424242",
  "appName" : "JCo-Demo",
  "appUrls" : {
    "getDependencies": "https://jco-demo-
p42424242.cfapps.eu10.hana.ondemand.com/callback/v1.0/dependencies",
    "onSubscription" : "https://jco-demo-
p42424242.cfapps.eu10.hana.ondemand.com/callback/v1.0/tenants/{tenantId}"
  }
}

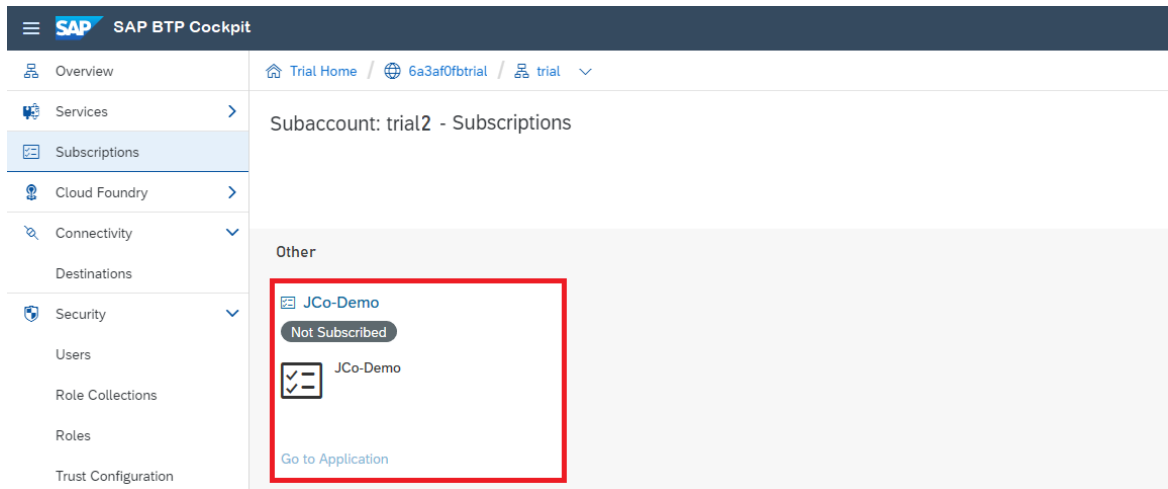
```

5. Choose **Next**, and select the sample application **jco-demo-p42424242** in the drop-down menu to assign the SAAS service to it.
6. Choose **Next**, insert an instance name, for example, **saas_jco**, and confirm the creation by pressing **Finish**.

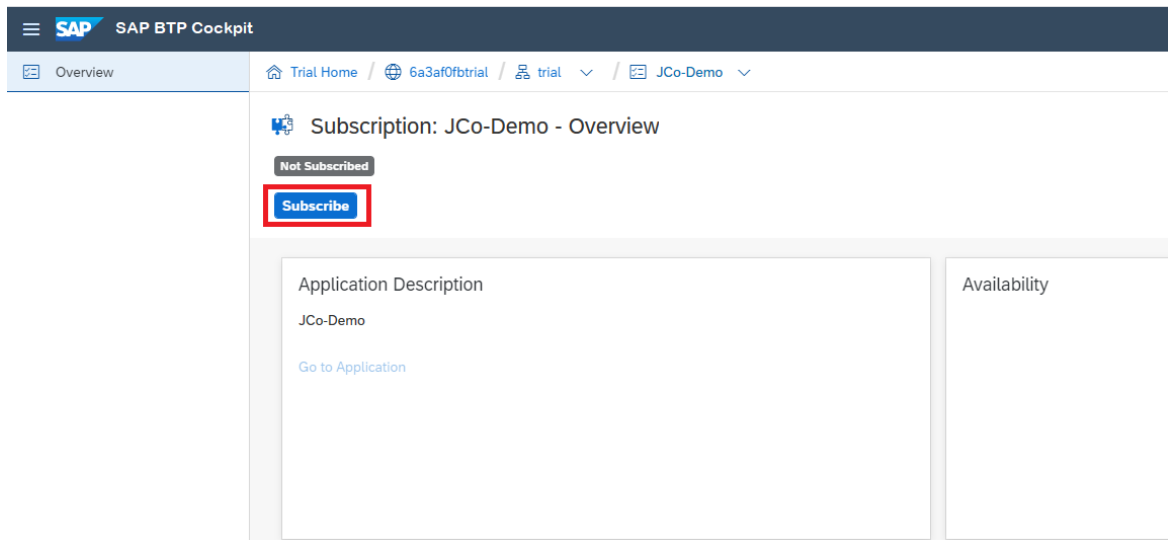
Back to [Steps \[page 323\]](#)

Subscribe to the Application

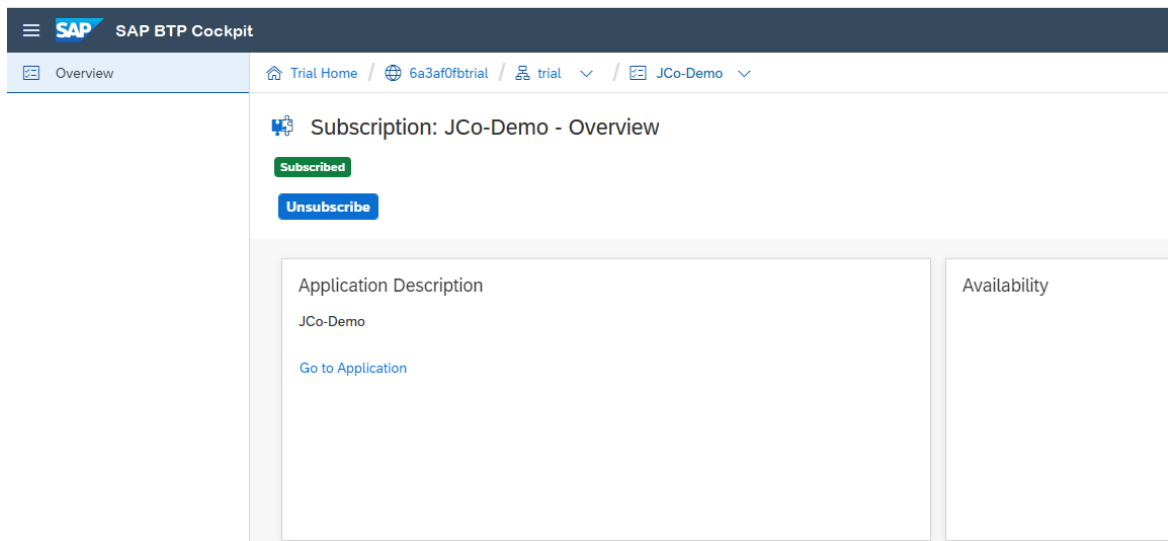
1. To subscribe the new application from a different subaccount, go to **Subscriptions** in the cockpit:



2. Click on *JCo-Demo*.
3. In the next window, choose *Subscribe*:



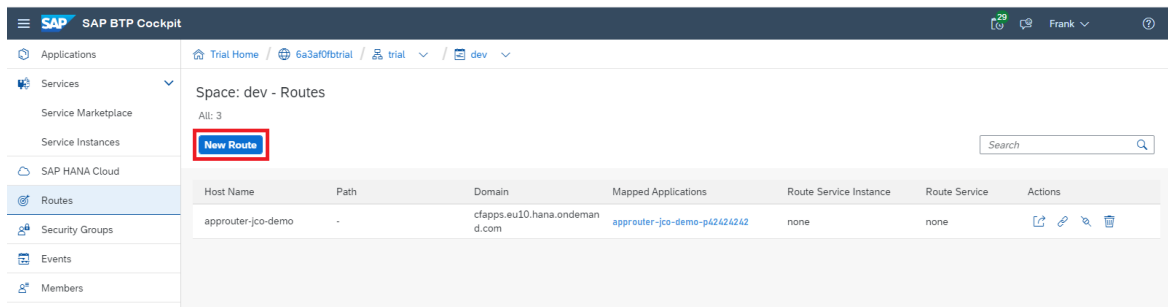
4. If the subscription was successful, your window should look like that:



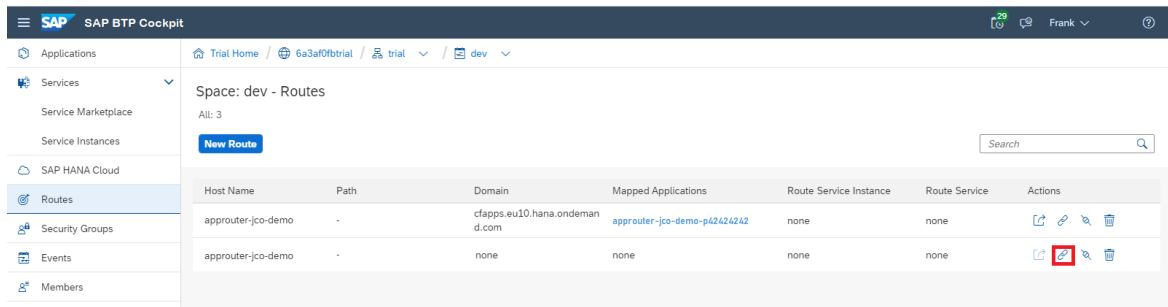
Back to [Steps \[page 323\]](#)

Create a New Route

1. To call the application with a new tenant, you must create a new route (URL). In the cockpit, choose **Routes** **New Route**:



2. For **<Domain>**, select the landscape your application is deployed in (e.g. `cfapps.eu10.hana.ondemand.com`).
3. For **<Host Name>**, choose the tenant-specific link. In our example it would be `<tenant-subdomain-name>.approuter-jco-demo-p42424242`.
4. Choose **Save**.
5. Choose **Map Route** of the newly created route where the field **<Mapped Applications>** is empty (value `none`):



6. Select the approuter application and choose **Save**.
7. Congratulations, you are done, now you are able to call the sample application with this newly created route from the other subaccount!

Back to [Steps \[page 323\]](#)

1.1.4.3.1.4 Configure Principal Propagation for RFC

Enable single sign-on (SSO) via RFC by forwarding the identity of cloud users from the Cloud Foundry environment to an on-premise system.

Prerequisites

You have set up the Cloud Connector and the relevant backend system for principal propagation. For more information, see [Configuring Principal Propagation \[page 420\]](#).

Procedure

- Make sure your RFC destination is configured with `jco.destination.auth_type=PrincipalPropagation`. For more information, see [User Logon Properties \[page 158\]](#).
- For your Java application, use an application router to forward a user token which is used by the Java Connector (JCo) for principal propagation. For more information, see [Set Up an Application Router \[page 295\]](#).

1.1.5 Security

Find an overview of recommended security measures for SAP BTP Connectivity.

Topic	More Information
Enable single sign-on by forwarding the identity of cloud users to a remote system or service.	Principal Propagation [page 168]
Set up and run the Cloud Connector according to the highest security standards.	Security Guidelines [page 711]

More Information

[SAP BTP Security Recommendations](#) collects information that lets you secure the configuration and operation of SAP BTP services in your landscape.

1.1.6 Monitoring and Troubleshooting

Find information on monitoring and troubleshooting for SAP BTP Connectivity.

Getting Support

If you encounter an issue with this service, we recommend to follow the procedure below:

Check Platform Status

Check the availability of the platform at [SAP BTP Status Page](#) .

For more information about selected platform incidents, see [Root Cause Analyses](#).

Check Guided Answers

In the SAP Support Portal, check the [Guided Answers](#) section for SAP BTP. You can find solutions for general SAP BTP issues as well as for specific services there.

Contact SAP Support

You can report an incident or error through the [SAP Support Portal](#) .

To find the relevant component for your for SAP BTP Connectivity incident, see [Connectivity Support \[page 869\]](#) (section *SAP Support Information*).

When submitting the incident, we recommend including the following information:

- Region information (for example: Canary, EU10, US10)
- Subaccount technical name
- The URL of the page where the incident or error occurs
- The steps or clicks used to replicate the error
- Screenshots, videos, or the code entered

More Information

Topic	More Information
Monitor the Cloud Connector from the SAP BTP cockpit and from the Cloud Connector administration UI.	Monitoring [page 660]
Troubleshoot connection problems and view different types of logs and traces in the Cloud Connector.	Troubleshooting [page 694]
Detailed support information for SAP Connectivity service and the Cloud Connector.	Connectivity Support [page 869]

1.1.7 Resilience Recommendations

Improve resilience of your SAP BTP applications.

While SAP strives to ensure the highest possible availability of the provided services, true resilience actually is a two-way collaboration between client and server. Thus, it is important to implement any applications using the Connectivity and/or Destination services (or any other services for that matter) in a resilient way. This page gives suggestions on measures to take in order to endure short disruptions, network issues, slowdowns or other abnormal situations that might arise. By doing this, these application can withstand such disruptions, ensuring business continuity in the face of underlying issues in the platform.

Note

When using client libraries like the [BTP security library](#), the [Cloud SDK](#), and so on, many of these recommendations may already be included. However, we recommend that you double-check these features, as they might require additional configuration.

Caching

Caching is an important pillar of resilience. It is the act of storing data (acquired from an external resource or as a result of a heavy computation) for future use. Caching operations must be done within a reasonable timespan to avoid the risk of data becoming invalid or outdated. So, caching lets you reduce the number of risky or heavy operations that go over the network towards an external resource. By doing this, the risk of failure is reduced, and caching additionally improves the performance of the application.

Note

Be careful what kind of data you cache and for how long. It is especially important to consider the handling of sensitive data (personal data, security objects, and so on).

Caution

Make sure you limit the size of the caches to avoid memory issues. There are different options when the maximum number of entities in the cache is reached. For example, you can drop the oldest entity from the cache in favor of the new one.

Token Caching

Caching access tokens to the services is highly recommended for the period of their validity. Access tokens to the service include the timestamp on which the token is no longer valid (as per JWT specification). Thus the tokens can be reused at any time before said timestamp. Keep in mind that tokens are issued for the specific clients and in the context of the specific tenant. Thus this should be taken into account when designing the caching mechanism.

Destination/Certificate Caching

The entities of the Destination service (be it destination configurations or certificate configuration) can also be cached. This includes both responses for single entities, as well as the list of entities responses. The drawback of this is that any changes in the configurations will not become immediately available to the application, thus a reasonable caching time (3-5 minutes is usually a good choice) must be set.

If you do not expect time critical changes in the destination configuration, you can set the caching time even higher (for example, an hour). Also, we recommend that you refresh the cache on-demand instead of calling the Destination service every 3-5 minutes.

By caching applications, they can withstand temporary issues with the service or the network. We also recommend that in cases where the cache has expired but retrieval of the updated entities does not succeed to still use the cached value as a further resilience measure.

Retry Logic

When requests (whether it is the token call, the service call or the business call) fail, it is always a good idea to retry the call over a reasonable time. It also makes sense to have some delay between retries, for example, at least 100 milliseconds. You can even make this delay increase with each subsequent retry. By doing this, you can handle short intermittent issues without having business impact.

Pagination

The APIs of the Destination service that return multiple entities support pagination. We highly recommended that you make use of pagination if you use these APIs and have a high number of entities (more than 100 entities). By doing this, you ease the pressure on the service but also make processing lighter on client side as the list can be handled in batches. See more about pagination in the documentation of the [Destination Service REST API \[page 80\]](#).

Timeouts

Connect and read timeouts help you keep your application resources from getting stuck in abnormally long processing. In some cases, one attempt to get data might get stuck, but abandoning it and trying again might immediately succeed.

We recommend you use appropriate *connect* and *read* timeouts when communicating with the services. The *connect* timeout is recommended to be on the low end - a couple of seconds. The *read* timeout would depend on the semantic of the call being made. A call to the Destination service REST API would not require a high read timeout. Around half a minute would be sufficient. The same is true for token retrieval calls. For business calls, including calls to on-premise systems via the Connectivity service, the timeout would be based on the type of the call and is heavily scenario-specific.

Circuit Breaker

The circuit-breaker architecture pattern can be a powerful tool for handling a misbehaving dependency. It reduces the effort done by your application when communicating to a service which is identified as not working or unstable, while waiting for the dependency to come back online.

For more information on this pattern, see <https://learn.microsoft.com/en-us/azure/architecture/patterns/circuit-breaker> .

Cloud Connector High Availability

See [High Availability Setup \[page 651\]](#).

1.1.8 On-Premise Connectivity in the Kyma Environment

The *connectivity-proxy* Kyma module enables secure tunneling between the Kyma environment and on-premise systems. It supports both *on-premise-to-cloud* and *cloud-to-on-premise* scenarios. It requires the *btp-operator* module and *Istio sidecar proxy injection* for proper functionality.

Prerequisites

- The service plan "*connectivity_proxy*" of the "*connectivity*" service is assigned to your subaccount as an entitlement.
For more information, see [Configure Entitlements and Quotas for Subaccounts](#).

Note

For subaccounts created after the February 15, 2024, this entitlement is assigned automatically.

- You have added the *btp-operator* module.
For more information, see [Add and Delete a Kyma Module](#).

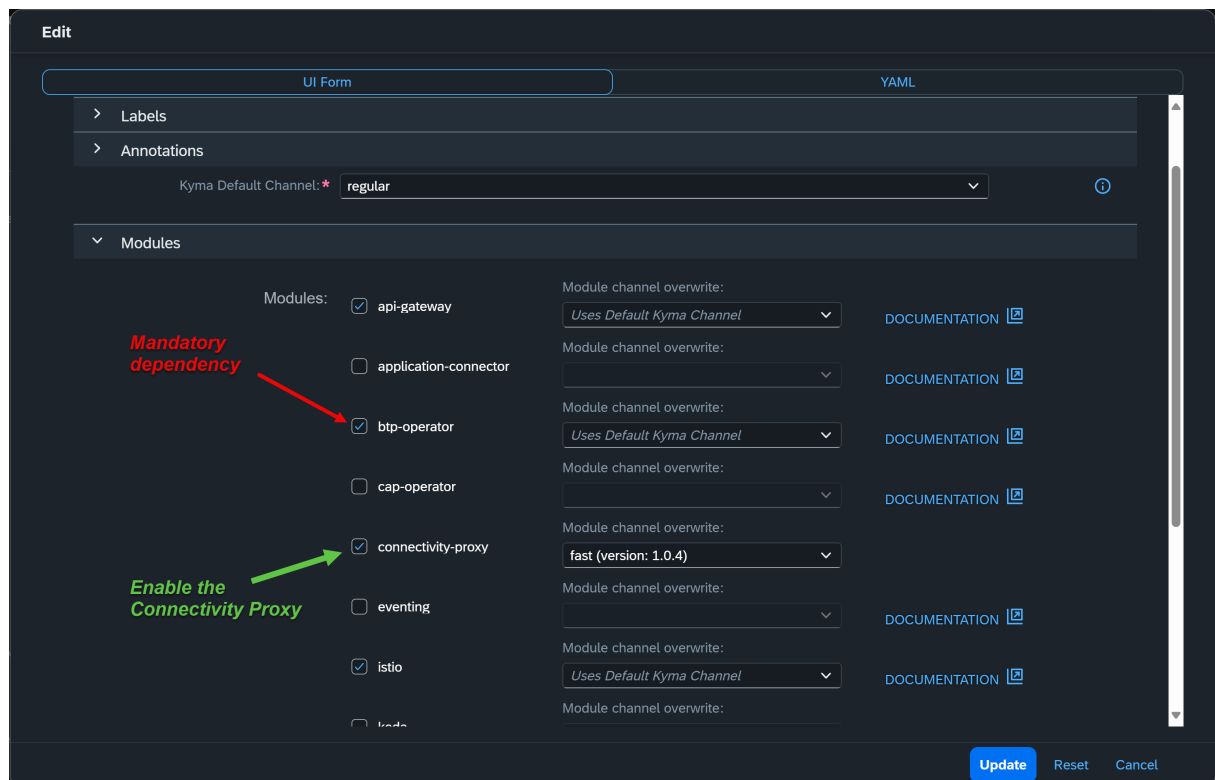
Context

The *connectivity-proxy* Kyma module installs the needed components to establish a secure tunnel between the Kyma environment and the systems in your on-premise network, exposed via [Cloud Connector \[page 343\]](#). The

module is based on the [Connectivity Proxy for Kubernetes \[page 732\]](#). It supports both *on-premise-to-cloud* and *cloud-to-on-premise* scenarios.

Enable the Connectivity Proxy module

The connectivity proxy is a standard Kyma module. You can enable the module as described in [Add and Delete a Kyma Module](#).



The Kubernetes resource that facilitates this installation in the background is a `ConnectivityProxy` resource. This is a custom resource, defined by the module. It holds the configuration for the connectivity proxy components.

For more information, see [Configuration Guide \[page 772\]](#).

Note

Not all configuration options of the connectivity proxy are supported by the Kyma module.

See [Limitations \[page 342\]](#) below for more details.

```
1 apiVersion: connectivityproxy.sap.com/v1
2 kind: ConnectivityProxy
3 spec:
4   config:
5     highAvailabilityMode: 'off'
6   integration:
7     auditlog:
8       mode: console
9     connectivityService:
10      serviceCredentialsKey: service_key
11 multiRegionMode:
12   enabled: false
13 servers:
14   businessDataTunnel:
15     externalHost: auto-provision
16     externalPort: 443
17   proxy:
18     http:
19       enableProxyAuthorization: false
20       enabled: true
21       port: 20003
22     rfcAndldap:
23       enableProxyAuthorization: false
24       enabled: true
25       port: 20001
26     socks5:
27       enableProxyAuthorization: false
28       enabled: true
29       port: 20004
```

Update Reset Cancel

⚠ Caution

Do not create additional resources of type `ConnectivityProxy`. Only one connectivity proxy installation per cluster is supported.

Result

The module will be enabled and will result in an installation of the [Connectivity Proxy for Kubernetes \[page 732\]](#) and its supporting workloads.

As part of this installation, a `ServiceInstance` and a `ServiceBinding` resource will be created in the cluster. These are *BTP Operator* resources which result in the creation of a *Connectivity* service instance with service plan `connectivity_proxy`, and in a service binding in your subaccount. This is needed to pair the connectivity proxy with the SAP BTP Connectivity service.

For more information, see [Connectivity Service \[page 753\]](#).

⚠ Caution

Do not interact with the created service instance and service binding, as they are only meant for usage by the connectivity proxy. Any modification may cause a loss of functionality.

The installation also exposes the public facing endpoint of the connectivity proxy via the built-in Istio Ingress controller (TLS certificates are generated automatically during this process). This endpoint is used by the Cloud Connector to reach the proxy.

The `Ingress` endpoint is propagated transparently for *cloud-to-on-premise* scenarios. For *on-premise-to-cloud* connections, you must look it up manually. You can find it in the created `Gateway` resource of the `kyma-system` namespace.

Capabilities

If no settings are modified, the following specifics and features are enabled in the `connectivity_proxy` Kyma module:

- All proxy servers are enabled (HTTP, RFC, LDAP, SOCKS5).
- The [Service Channels: On-Premise-to-Cloud Connectivity \[page 756\]](#) feature is enabled.
- Operational mode is [single tenant, trusted \[page 739\]](#).
- [Automatic Pickup on Resource Changes \[page 755\]](#) is enabled.

Limitations

The following features of the connectivity proxy are not available via the `connectivity-proxy` Kyma module:

- [High Availability \[page 749\]](#)
- [Multi-Region Mode \[page 760\]](#)
- Support for Ingress controllers other than Istio
- An externally exposed health check endpoint

Consume the Connectivity Proxy

As part of the installation, a special `ConfigMap` will be created in the `kyma-system` namespace, called `connectivity-proxy-info`. It contains the host of the connectivity proxy and all its proxy ports.

```
1 kind: ConfigMap
2 apiVersion: v1
3 metadata:
4   name: connectivity-proxy-info
5   namespace: kyma-system
6 data:
7   onpremise_proxy_host: connectivity-proxy.kyma-system.svc.cluster.local
8   onpremise_proxy_http_port: '20003'
9   onpremise_proxy_ldap_port: '20001'
10  onpremise_proxy_port: '20003'
11  onpremise_proxy_rfc_port: '20001'
12  onpremise_socks5_proxy_port: '20004'
13
```

Update Reset Cancel

To use the connectivity proxy, you must set the appropriate host and port as proxy configuration.

For more information, see [Using the Connectivity Proxy \[page 788\]](#).

⚠ Caution

Every workload using the connectivity proxy to call the on-premise system must have the Istio sidecar proxy injection enabled. Otherwise, the connectivity proxy does not work correctly.

1.2 Cloud Connector

Learn more about the Cloud Connector: features, scenarios and setup.

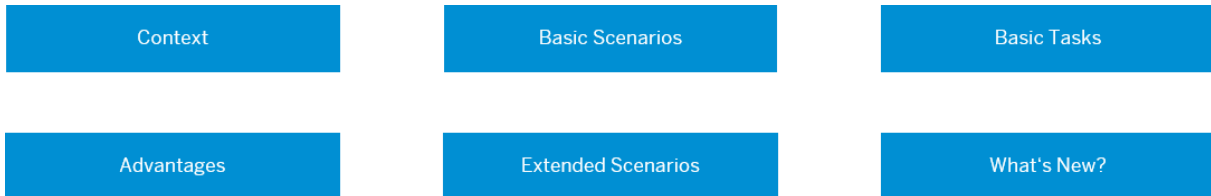
📄 Note

This documentation refers to SAP BTP, Cloud Foundry environment. If you are looking for information about the Neo environment, see [Connectivity for the Neo Environment](#).

Content

In this Topic

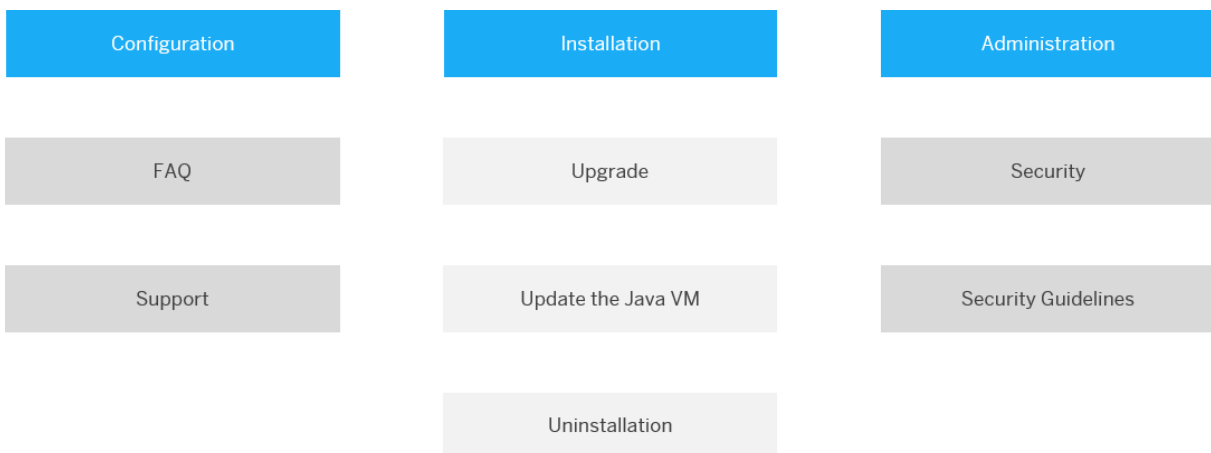
Hover over the elements for a description. Click an element for more information.



- [Context \[page 345\]](#)
- [Basic Scenarios \[page 346\]](#)
- [Basic Tasks \[page 348\]](#)
- [Advantages \[page 345\]](#)
- [Extended Scenarios \[page 348\]](#)
- [What's New? \[page 349\]](#)

In this Guide

Hover over the elements for a description. Click an element for more information.



- [Configuration \[page 387\]](#)
- [Installation \[page 349\]](#)
- [Administration \[page 628\]](#)
- [Frequently Asked Questions \[page 719\]](#)
- [Upgrade \[page 715\]](#)

- [Security \[page 703\]](#)
- [Update the Java VM \[page 716\]](#)
- [Uninstallation \[page 718\]](#)
- [Connectivity Support \[page 869\]](#)
- [Security Guidelines \[page 711\]](#)

Context

The Cloud Connector:

- Serves as a link between SAP BTP applications and on-premise systems.
 - Combines an easy setup with a clear configuration of the systems that are exposed to the SAP BTP.
 - Lets you use existing on-premise assets without exposing the entire internal landscape.
- Runs as on-premise agent in a secured network.
 - Acts as a reverse invoke proxy between the on-premise network and SAP BTP.
- Provides fine-grained control over:
 - On-premise systems and resources that can be accessed by cloud applications.
 - Cloud applications using the Cloud Connector.
- Lets you use the features that are required for business-critical enterprise scenarios.
 - Recovers broken connections automatically.
 - Provides audit logging of inbound traffic and configuration changes.
 - Can be run in a high-availability setup.

⚠ Caution

The Cloud Connector must not be used to connect to products other than SAP BTP or S/4HANA Cloud.

Back to [Content \[page 344\]](#)

Advantages

Compared to the approach of opening ports in the firewall and using reverse proxies in the DMZ to establish access to on-premise systems, the Cloud Connector offers the following benefits:

- You don't need to configure the on-premise firewall to allow external access from SAP BTP to internal systems. For allowed outbound connections, no modifications are required.
- The Cloud Connector supports HTTP as well as additional protocols. For example, the RFC protocol supports native access to ABAP systems by invoking function modules.
- You can use the Cloud Connector to connect on-premise databases or BI tools to SAP HANA databases in the cloud.

- The Cloud Connector lets you propagate the identity of cloud users to on-premise systems in a secure way.
- Easy installation and configuration, which means that the Cloud Connector comes with a low TCO and is tailored to fit your cloud scenarios.
- SAP provides standard support for the Cloud Connector.

Back to [Content \[page 344\]](#)

Basic Scenarios

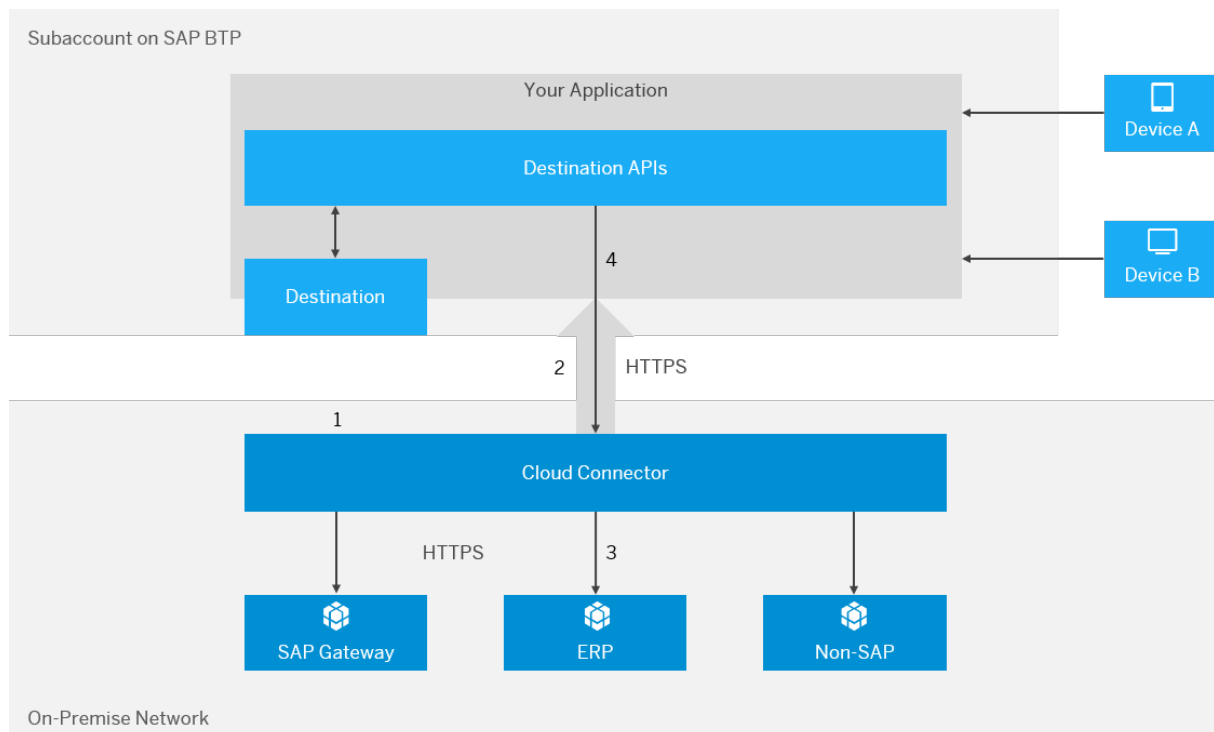
[Connecting Cloud Applications to On-Premise Systems \[page 346\]](#)

[Connecting On-Premise Database Tools to SAP HANA Databases \[page 347\]](#)

Note

This section refers to the Cloud Connector installation in a standard on-premise network. Find setup options for other system environments in [Extended Scenarios \[page 348\]](#).

Connecting Cloud Applications to On-Premise Systems



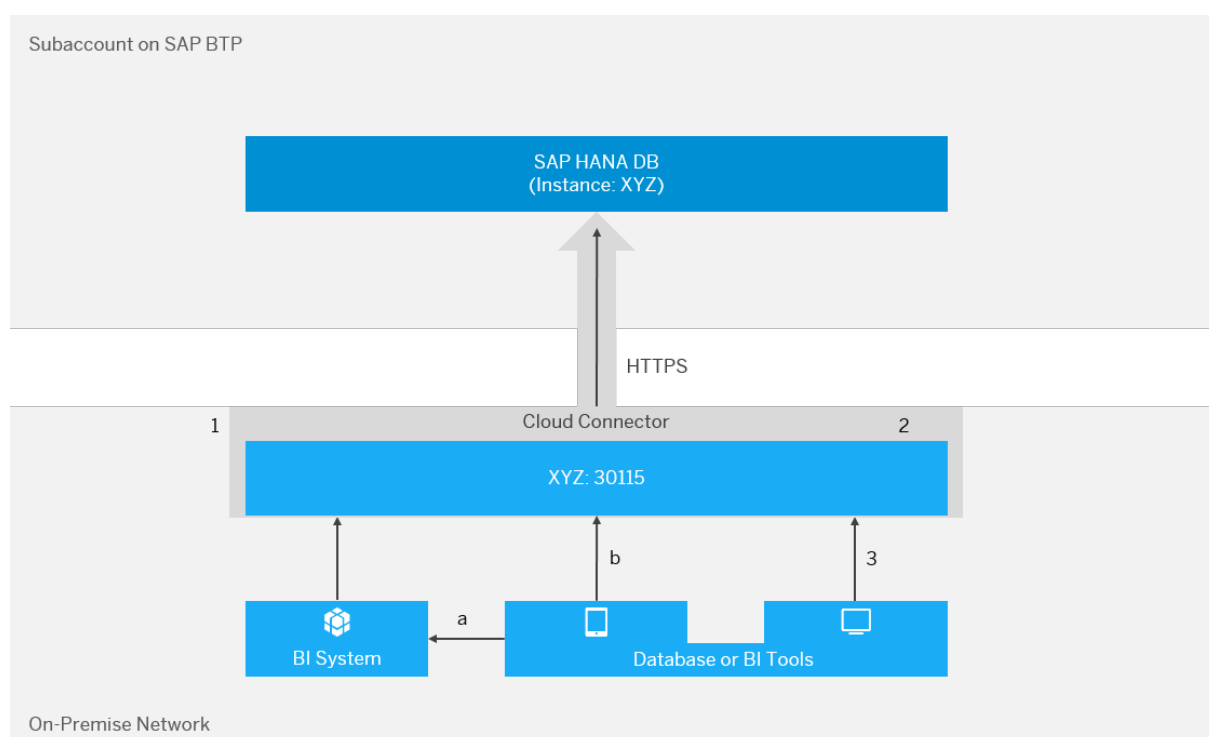
1. Install the Cloud Connector: [Installation \[page 349\]](#)
2. Set up the connection between Cloud Connector, back-end system and your SAP BTP subaccount : [Initial Configuration \[page 388\]](#), [Managing Subaccounts \[page 401\]](#)

3. Allow your cloud application to access a back-end system on the intranet: [Configure Access Control \[page 456\]](#)
4. Connect your cloud application to an on-premise system: [Consuming the Connectivity Service \[page 214\]](#) (Cloud Foundry environment)

Back to [Basic Scenarios \[page 346\]](#)

Back to [Content \[page 344\]](#)

Connecting On-Premise Database Tools to SAP HANA Databases



1. Install and configure the Cloud Connector: [Installation \[page 349\]](#), [Initial Configuration \[page 388\]](#), [Managing Subaccounts \[page 401\]](#)
2. Access HANA databases on SAP BTP: [Configure a Service Channel for an SAP HANA Database](#)
3. Connect on-premise database or BI tools to a HANA database on SAP BTP: [Connect DB Tools to SAP HANA via Service Channels \[page 609\]](#)

Note

You can use service channels also for other purposes:

- Connect to a virtual machine on SAP BTP.
- Configure an RFC connection from your on-premise system to S/4HANA Cloud.

See [Using Service Channels \[page 608\]](#).

Back to [Basic Scenarios \[page 346\]](#)

Back to [Content \[page 344\]](#)

Extended Scenarios

Besides the standard setup: *SAP BTP - Cloud Connector - on-premise system/network*, you can also use the Cloud Connector to connect SAP BTP applications to other cloud-based environments, as long as they are operated in a way that is comparable to an on-premise network from a functional perspective. This is particularly true for infrastructure (IaaS) hosting solutions.

Here's an overview of all environments in which you can or cannot set up the Cloud Connector:

Cloud Connector	Environment	Examples
Can be set up in:	Customer on-premise network (see Basic Scenarios [page 346])	SAP ERP, SAP S/4HANA
	SAP Hosting	SAP HANA Enterprise Cloud (HEC)
	Third-party IaaS providers (hosting)	Amazon Web Services (AWS), Microsoft Azure, Google Cloud
Cannot be set up in:	SAP SaaS solutions	SAP SuccessFactors, SAP Concur, SAP Ariba
	SAP cloud-based enterprise solutions	SAP S/4HANA Cloud, SAP C/4HANA
	Third-party PaaS providers	AWS, Azure, Google Cloud
	Third-party SaaS providers	

Note

Within extended scenarios that allow a Cloud Connector setup, special procedures may apply for configuration. If so, they are mentioned in the corresponding configuration steps.

Back to [Content \[page 344\]](#)

Basic Tasks

The following steps are required to connect the Cloud Connector to your SAP BTP subaccount:

- Install the Cloud Connector: [Installation \[page 349\]](#)
- Perform the initial configuration for the Cloud Connector: [Initial Configuration \[page 388\]](#)
- Register the Cloud Connector for your SAP BTP subaccount: [Managing Subaccounts \[page 401\]](#)

Back to [Content \[page 344\]](#)

What's New?

Follow the SAP BTP [Release Notes](#) to stay informed about Cloud Connector and Connectivity updates.

Back to [Content \[page 344\]](#)

Related Information

[Installation \[page 349\]](#)

[Configuration \[page 387\]](#)

[Administration \[page 628\]](#)

[Security \[page 703\]](#)

[Upgrade \[page 715\]](#)

[Update the Java VM \[page 716\]](#)

[Uninstallation \[page 718\]](#)

[Frequently Asked Questions \[page 719\]](#)

[REST APIs \[page 728\]](#)

1.2.1 Installation

Choose a procedure to install the Cloud Connector on your operating system.

Portable Version vs. Installer Version

On **Microsoft Windows** and **Linux**, two installation modes are available: a `portable` version and an `installer` version. On **Mac OS X**, only the `portable` version is available.

- `portable` version: can be installed easily, by extracting a compressed archive into an empty directory. It does not require administrator or root privileges for the installation, and you can run multiple instances on the same host.
Restrictions:
 - You cannot run it in the background as a Windows Service or Linux daemon (with automatic start capabilities at boot time).
 - The portable version does not support an automatic upgrade procedure. To update a portable installation, you must delete the current one, extract the new version, and then re-do the configuration.
 - Portable versions are meant for non-productive scenarios only.
 - The environment variable `JAVA_HOME` is relevant when starting the instance, and therefore must be set properly.

- `Installer` version: requires administrator or root permissions for the installation and can be set up to run as a Windows service or Linux daemon in the background. You can upgrade it easily, retaining all the configuration and customizing.

Note

We strongly recommend that you use this variant for a productive setup.

Caution

Cloud Connector is based on Tomcat.

Tomcat is a well-known and well-documented server, whose configuration can be adapted to one's own needs. However, we strongly recommend that you **do not modify** its configuration files like `conf\server.xml` with a text editor, because the Cloud Connector is making assumptions about the content of those files.

If you still want to do custom modifications, you do so at your own risk. In this case, we can no longer guarantee that the Cloud Connector keeps working as expected. For any issues that can be traced back to such changes, we cannot provide support, in particular, if such changes cause trouble during upgrade to a newer version.

Prerequisites

- There are some general prerequisites you must fulfill to successfully install the Cloud Connector, see [Prerequisites \[page 351\]](#).
- For OS-specific requirements and procedures, see section **Tasks** below.

Tasks

- [Installation on Microsoft Windows OS \[page 375\]](#)
- [Installation on Linux OS \[page 378\]](#)
- [Installation on Apple macOS \[page 381\]](#)

Related Information

[Sizing Recommendations \[page 369\]](#)

[Recommendations for Secure Setup \[page 383\]](#)

[Uninstallation \[page 718\]](#)

1.2.1.1 Prerequisites

Prerequisites for successful installation of the Cloud Connector.

Content

Section	Description
Connectivity Restrictions [page 351]	General information about SAP BTP and connectivity restrictions.
Hardware [page 351]	Hardware prerequisites for a physical or virtual machine.
Software [page 352]	Required software download and installation.
JDKs [page 352]	Java Development Kit (JDK) versions that you can use.
Product Availability Matrix [page 353]	Availability of operating systems/versions for specific Cloud Connector versions.
Network [page 354]	Required Internet connection to SAP BTP hosts per region.

Note

For additional system requirements, see also [System Requirements \[page 367\]](#).

Connectivity Restrictions

For general information about SAP BTP restrictions, see [Prerequisites and Restrictions](#).

For specific information about all Connectivity restrictions, see [Connectivity: Restrictions \[page 6\]](#).

Back to [Content \[page 351\]](#)

Hardware

Hardware prerequisites, physical or virtual machine:

	Minimum	Recommended
CPU	Single core 3 GHz, x86-64 architecture compatible	Dual core 2 GHz, x86-64 architecture compatible

	Minimum	Recommended
Memory (RAM)	2 GB	4 GB
Free disk space	3 GB	20 GB

Back to [Content \[page 351\]](#)

Software

- You have downloaded the Cloud Connector installation archive from [SAP Development Tools for Eclipse](#).
- A full JDK must be installed. Lightweight JRE installations are not sufficient. You can download a fitting up-to-date SAP JVM from [SAP Development Tools for Eclipse](#).

⚠ Caution

Do not use [Apache Portable Runtime \(APR\)](#) on the system on which you use the Cloud Connector. If you cannot avoid this restriction and want to use APR at your own risk, you must manually adapt the `server.xml` configuration file in directory `<scv_installation_folder>/conf`. To do so, follow the steps in [HTTPS port configuration](#) for APR.

Back to [Content \[page 351\]](#)

JDKs

JDK	Version	Cloud Connector Version
SAP JVM 64-bit (recommended)	7	2.x up to 2.12.2
	8	2.7.2 and higher
Oracle JDK 64-bit	7	2.x up to 2.12.2
	8	2.7.2 and higher
SAP Machine 64-bit	11	2.14.0 and higher
SAP Machine 64-bit	17	2.15.0 and higher

Back to [Content \[page 351\]](#)

Product Availability Matrix

Operating System Version	Architecture	Cloud Connector Version
Windows 7, Windows Server 2008 R2	x86_64	2.x
SUSE Linux Enterprise Server 11, Red Hat Enterprise Linux 6	x86_64	2.x
Mac OS X 10.7 (Lion), Mac OS X 10.8 (Mountain Lion)	x86_64	2.x
Windows 8.1, Windows Server 2012, Windows Server 2012 R2	x86_64	2.5.1 and higher
SUSE Linux Enterprise Server 12, Red Hat Enterprise Linux 7	x86_64	2.5.1 and higher
Mac OS X 10.9 (Mavericks), Mac OS X 10.10 (Yosemite)	x86_64	2.5.1 and higher
Windows 10	x86_64	2.7.2 and higher
Mac OS X 10.11 (El Capitan)	x86_64	2.8.1 and higher
Windows Server 2016	x86_64	2.9.1 and higher
Windows Server 2019, Mac OS X 10.12 (Sierra), Mac OS X 10.13 (High Sierra), Mac OS X 10.14 (Mojave)	x86_64	2.11.3 and higher
SUSE Linux Enterprise Server 15	x86_64	2.12.0 and higher
Red Hat Enterprise Linux 8	x86_64	2.12.2 and higher
SUSE Linux Enterprise Server 12, SUSE Linux Enterprise Server 15, Red Hat Enterprise Linux 7, Red Hat Enterprise Linux 8	ppc64le	2.13.0 and higher
Windows Server 2022, macOS 10.15 (Catalina)	x86_64	2.14.0 and higher
Windows 11, Red Hat Enterprise Linux 9	x86_64	2.15.0 and higher
Red Hat Enterprise Linux 9	ppc64le	2.15.0 and higher
Oracle Linux 8	x86_64	2.15.2 and higher
Oracle Linux 9, macOS 11 (Big Sur), macOS 12 (Monterey), macOS 13 (Ventura)	x86_64	2.16.0 and higher
macOS 11 (Big Sur), macOS 12 (Monterey), macOS 13 (Ventura)	aarch64	2.16.0 and higher

[Back to Content \[page 351\]](#)

Network

You must have Internet connection at least to the following Connectivity service hosts (depending on the region), to which you can connect your Cloud Connector. All connections to the hosts are TLS-based and connect to port 443.

→ Remember

For some solutions of the BTP portfolio, you must include additional hosts to set up an on-premise connectivity scenario with the Cloud Connector. This applies, for example, to: SAP Data Intelligence, SAP HANA Cloud, Business Application Studio, and SAP Build Apps. Check the respective solution documentation for details.

📘 Note

For general information on IP ranges per region, see [Regions](#) (Cloud Foundry and ABAP environment) or [Regions and Hosts Available for the Neo Environment](#). Find detailed information about the region status and planned network updates on [Platform Updates and Notifications](#).

[Cloud Foundry Environment \[page 354\]](#)

[ABAP Environment \[page 363\]](#)

[Neo Environment \[page 363\]](#)

[Trial \[page 365\]](#)

Region (Region Host)	Hosts	IP Address
Cloud Foundry Environment		
<div style="border: 1px solid #ccc; padding: 5px;"> <p>📘 Note</p> <p>In the Cloud Foundry environment, IPs are controlled by the respective IaaS provider - Amazon Web Services (AWS), Microsoft Azure (Azure), or Google Cloud. IPs may change due to network updates on the provider side. Any planned changes will be announced at least 4 weeks before they take effect. See also Regions.</p> </div>		
Europe (Frankfurt) - AWS	connectivitynotification.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223
(cf.eu10.hana.ondemand.com)		Additional IP addresses (valid after March 31, 2024): 18.159.31.22, 3.69.186.98, 3.77.195.119
<i>Enterprise & Trial</i>		
<div style="border: 1px solid #ccc; padding: 5px;"> <p>⚠ Caution</p> <p>Additional IP addresses were added to this region.</p> </div>		

Region (Region Host)	Hosts	IP Address
Action: If you restrict system access by <i>allowlisting IPs</i> in firewall rules, make sure you update your configuration as soon as possible. The additional IP addresses will be used after March 31, 2024.	connectivitycertsigning.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223 Additional IP addresses (valid after March 31, 2024): 18.159.31.22, 3.69.186.98, 3.77.195.119
	connectivitytunnel.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223 Additional IP addresses (valid after March 31, 2024): 18.159.31.22, 3.69.186.98, 3.77.195.119
	connectivitytunnel.cf.eu10-002.hana.ondemand.com	3.64.227.236, 3.126.229.22, 18.193.180.19 Additional IP addresses (valid after March 31, 2024): 18.153.123.11, 3.121.37.195, 3.73.215.90
	connectivitytunnel.cf.eu10-003.hana.ondemand.com	3.127.77.3, 3.64.196.58, 18.156.151.247 Additional IP addresses (valid after March 31, 2024): 18.197.252.154, 3.79.137.29, 52.58.93.50
	connectivitytunnel.cf.eu10-004.hana.ondemand.com	3.65.185.47, 3.70.38.218, 18.196.206.8 Additional IP addresses (valid after March 31, 2024): 3.73.109.100, 3.73.8.210, 52.59.18.183

Region (Region Host)	Hosts	IP Address
Europe (Frankfurt) - AWS (cf.eu11.hana.ondemand.com)	connectivitynotification.cf.eu11.hana.ondemand.com	3.124.207.41, 18.157.105.117, 18.156.209.198 Additional IP addresses (valid after March 31, 2024): 3.66.26.249, 3.72.216.204, 3.74.99.245
	connectivitycertsigning.cf.eu11.hana.ondemand.com	3.124.207.41, 18.157.105.117, 18.156.209.198 Additional IP addresses (valid after March 31, 2024): 3.66.26.249, 3.72.216.204, 3.74.99.245
	connectivitytunnel.cf.eu11.hana.ondemand.com	3.124.207.41, 18.157.105.117, 18.156.209.198 Additional IP addresses (valid after March 31, 2024): 3.66.26.249, 3.72.216.204, 3.74.99.245
Europe (Netherlands) - Azure (cf.eu20.hana.ondemand.com)	connectivitynotification.cf.eu20.hana.ondemand.com	40.119.153.88
	connectivitycertsigning.cf.eu20.hana.ondemand.com	40.119.153.88
	connectivitytunnel.cf.eu20.hana.ondemand.com	40.119.153.88
	connectivitytunnel.cf.eu20-001.hana.ondemand.com	20.82.83.59
Europe (Frankfurt) - Google Cloud (cf.eu30.hana.ondemand.com)	connectivitynotification.cf.eu30.hana.ondemand.com	35.198.143.110
	connectivitycertsigning.cf.eu30.hana.ondemand.com	35.198.143.110
	connectivitytunnel.cf.eu30.hana.ondemand.com	35.198.143.110

⚠ Caution

Additional IP addresses were added to this region.

Action:

If you restrict system access by *allowlisting* IPs in firewall rules, make sure you update your configuration as soon as possible. The additional IP addresses will be used after March 31, 2024.

Region (Region Host)	Hosts	IP Address
US East (VA) - AWS (cf.us10.hana.ondemand.com) Enterprise & Trial	connectivitynotification.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211 Additional IP addresses (valid after March 31, 2024): 18.213.242.208, 3.214.110.153, 34.205.56.51
	connectivitycertsigning.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211 Additional IP addresses (valid after March 31, 2024): 18.213.242.208, 3.214.110.153, 34.205.56.51
	connectivitytunnel.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211 Additional IP addresses (valid after March 31, 2024): 18.213.242.208, 3.214.110.153, 34.205.56.51
	connectivitytunnel.cf.us10-001.hana.ondemand.com	3.220.114.17, 3.227.182.44, 52.86.131.53 Additional IP addresses (valid after March 31, 2024): 44.218.82.203, 44.219.57.163, 50.16.106.103

⚠ Caution

Additional IP addresses were added to this region.

Action:

If you restrict system access by *allowlisting* IPs in firewall rules, make sure you update your configuration as soon as possible. The additional IP addresses will be used after March 31, 2024.

Region (Region Host)	Hosts	IP Address
	connectivitytunnel.cf.us10-002.hana.ondemand.com	34.202.68.0, 54.234.152.59, 107.20.66.86
		Additional IP addresses (valid after March 31, 2024): 3.214.116.95, 54.144.230.36, 54.226.37.104
US West (WA) - Azure	connectivitynotification.cf.us20.hana.ondemand.com	40.91.120.100
(cf.us20.hana.ondemand.com)	connectivitycertsigning.cf.us20.hana.ondemand.com	40.91.120.100
	connectivitytunnel.cf.us20.hana.ondemand.com	40.91.120.100
US East (VA) - Azure	connectivitynotification.cf.us21.hana.ondemand.com	40.88.52.17
(cf.us21.hana.ondemand.com)	connectivitycertsigning.cf.us21.hana.ondemand.com	40.88.52.17
	connectivitytunnel.cf.us21.hana.ondemand.com	40.88.52.17
US Central (IA) - Google Cloud	connectivitynotification.cf.us30.hana.ondemand.com	35.184.169.79
(cf.us30.hana.ondemand.com)	connectivitycertsigning.cf.us30.hana.ondemand.com	35.184.169.79
	connectivitytunnel.cf.us30.hana.ondemand.com	35.184.169.79
Brazil (São Paulo) - AWS	connectivitynotification.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4, 54.232.179.204
(cf.br10.hana.ondemand.com)		Additional IP addresses (valid after March 31, 2024): 18.228.53.198, 52.67.149.240, 54.94.179.209
	connectivitycertsigning.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4, 54.232.179.204
		Additional IP addresses (valid after March 31, 2024): 18.228.53.198, 52.67.149.240, 54.94.179.209

⚠ Caution

Additional IP addresses were added to this region.

Action:

If you restrict system access by *allowlisting* IPs in firewall rules, make sure you update your configuration as soon as possible. The additional IP addresses will be used after March 31, 2024.

Region (Region Host)	Hosts	IP Address
	connectivitytunnel.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4, 54.232.179.204 Additional IP addresses (valid after March 31, 2024): 18.228.53.198, 52.67.149.240, 54.94.179.209
Japan (Tokyo) - AWS (cf.jp10.hana.ondemand.com)	connectivitynotification.cf.jp10.hana.ondemand.com	13.114.117.83, 3.114.248.68, 3.113.252.15 Additional IP addresses (valid after March 31, 2024): 18.178.155.134, 57.180.140.5, 57.180.145.179
<div style="border-left: 2px solid orange; padding-left: 10px;"> <p>⚠ Caution</p> <p>Additional IP addresses were added to this region.</p> <p>Action:</p> <p>If you restrict system access by <i>allowlisting</i> IPs in firewall rules, make sure you update your configuration as soon as possible. The additional IP addresses will be used after March 31, 2024.</p> </div>	connectivitycertsigning.cf.jp10.hana.ondemand.com	13.114.117.83, 3.114.248.68, 3.113.252.15 Additional IP addresses (valid after March 31, 2024): 18.178.155.134, 57.180.140.5, 57.180.145.179
	connectivitytunnel.cf.jp10.hana.ondemand.com	13.114.117.83, 3.114.248.68, 3.113.252.15 Additional IP addresses (valid after March 31, 2024): 18.178.155.134, 57.180.140.5, 57.180.145.179
	Japan (Tokyo) - Azure (cf.jp20.hana.ondemand.com)	connectivitynotification.cf.jp20.hana.ondemand.com
	connectivitycertsigning.cf.jp20.hana.ondemand.com	20.43.89.91
	connectivitytunnel.cf.jp20.hana.ondemand.com	20.43.89.91

Region (Region Host)	Hosts	IP Address
Australia (Sydney) - AWS (cf.ap10.hana.ondemand.com)	connectivitynotification.cf.ap10.hana.ondemand.com	13.236.220.84, 13.211.73.244, 3.105.95.184 Additional IP addresses (valid after March 31, 2024): 13.55.188.95, 3.105.212.249, 3.106.45.106
	connectivitycertsigning.cf.ap10.hana.ondemand.com	13.236.220.84, 13.211.73.244, 3.105.95.184 Additional IP addresses (valid after March 31, 2024): 13.55.188.95, 3.105.212.249, 3.106.45.106
	connectivitytunnel.cf.ap10.hana.ondemand.com	13.236.220.84, 13.211.73.244, 3.105.95.184 Additional IP addresses (valid after March 31, 2024): 13.55.188.95, 3.105.212.249, 3.106.45.106
Asia Pacific (Singapore) - AWS (cf.ap11.hana.ondemand.com)	connectivitynotification.cf.ap11.hana.ondemand.com	3.0.9.102,18.140.39.70, 18.139.147.53 Additional IP addresses (valid after March 31, 2024): 13.229.158.122, 18.140.228.217, 52.74.215.89
	connectivitycertsigning.cf.ap11.hana.ondemand.com	3.0.9.102, 18.140.39.70, 18.139.147.53 Additional IP addresses (valid after March 31, 2024): 13.229.158.122, 18.140.228.217, 52.74.215.89

⚠ Caution

Additional IP addresses were added to this region.

Action:

If you restrict system access by *allowlisting* IPs in firewall rules, make sure you update your configuration as soon as possible. The additional IP addresses will be used after March 31, 2024.

⚠ Caution

Additional IP addresses were added to this region.

Action:

If you restrict system access by *allowlisting* IPs in firewall rules, make sure you update your configuration as soon as possible. The additional IP ad-

Region (Region Host)	Hosts	IP Address
<p>dresses will be used after March 31, 2024.</p>	connectivitytunnel.cf.ap11.hana.ondemand.com	3.0.9.102, 18.140.39.70, 18.139.147.53 Additional IP addresses (valid after March 31, 2024): 13.229.158.122, 18.140.228.217, 52.74.215.89
	connectivitynotification.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12 Additional IP addresses (valid after March 31, 2024): 13.209.236.215, 43.201.194.105, 43.202.204.5
<p>Asia Pacific (Seoul) - AWS (cf.ap12.hana.ondemand.com)</p>	connectivitycertsigning.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12 Additional IP addresses (valid after March 31, 2024): 13.209.236.215, 43.201.194.105, 43.202.204.5
<p>⚠ Caution</p> <p>Additional IP addresses were added to this region.</p> <p>Action:</p> <p>If you restrict system access by <i>allowlisting</i> IPs in firewall rules, make sure you update your configuration as soon as possible. The additional IP addresses will be used after March 31, 2024.</p>	connectivitytunnel.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12 Additional IP addresses (valid after March 31, 2024): 13.209.236.215, 43.201.194.105, 43.202.204.5
	connectivitynotification.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12 Additional IP addresses (valid after March 31, 2024): 13.209.236.215, 43.201.194.105, 43.202.204.5
	connectivitycertsigning.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12 Additional IP addresses (valid after March 31, 2024): 13.209.236.215, 43.201.194.105, 43.202.204.5
<p>Australia (Sydney) - Azure (cf.ap20.hana.ondemand.com)</p>	connectivitynotification.cf.ap20.hana.ondemand.com	20.53.99.41
	connectivitycertsigning.cf.ap20.hana.ondemand.com	20.53.99.41
	connectivitytunnel.cf.ap20.hana.ondemand.com	20.53.99.41
<p>Singapore - Azure</p>	connectivitynotification.cf.ap21.hana.ondemand.com	20.184.61.122
	connectivitycertsigning.cf.ap21.hana.ondemand.com	20.184.61.122

Region (Region Host)	Hosts	IP Address
(cf.ap21.hana.ondemand.com) <i>Enterprise & Trial</i>	connectivitytunnel.cf.ap21.hana.ondemand.com	20.184.61.122
Canada (Montreal) - AWS (cf.ca10.hana.ondemand.com)	connectivitynotification.cf.ca10.hana.ondemand.com	3.98.102.153, 35.182.75.101, 35.183.74.34 Additional IP addresses (valid after March 31, 2024): 15.157.88.166, 3.98.202.222, 52.60.210.33
<div style="border: 1px solid orange; padding: 5px;"> <p>⚠ Caution</p> <p>Additional IP addresses were added to this region.</p> <p>Action:</p> <p>If you restrict system access by <i>allowlisting</i> IPs in firewall rules, make sure you update your configuration as soon as possible. The additional IP addresses will be used after March 31, 2024.</p> </div>	connectivitycertsigning.cf.ca10.hana.ondemand.com	3.98.102.153, 35.182.75.101, 35.183.74.34 Additional IP addresses (valid after March 31, 2024): 15.157.88.166, 3.98.202.222, 52.60.210.33
	connectivitytunnel.cf.ca10.hana.ondemand.com	3.98.102.153, 35.182.75.101, 35.183.74.34 Additional IP addresses (valid after March 31, 2024): 15.157.88.166, 3.98.202.222, 52.60.210.33
	connectivitynotification.cf.ch20.hana.ondemand.com	20.208.56.83
(cf.ch20.hana.ondemand.com)	connectivitycertsigning.cf.ch20.hana.ondemand.com	20.208.56.83
	connectivitytunnel.cf.ch20.hana.ondemand.com	20.208.56.83
	connectivitynotification.cf.in30.hana.ondemand.com	34.93.125.74
India (Mumbai) - Google Cloud (cf.in30.hana.ondemand.com)	connectivitycertsigning.cf.in30.hana.ondemand.com	34.93.125.74
	connectivitytunnel.cf.in30.hana.ondemand.com	34.93.125.74
	connectivitynotification.cf.cn40.platform.sapcloud.cn	139.224.7.71
China (Shanghai) - Alibaba Cloud (cf.cn40.platform.sapcloud.cn)	connectivitycertsigning.cf.cn40.platform.sapcloud.cn	139.224.7.71
	connectivitytunnel.cf.cn40.platform.sapcloud.cn	139.224.7.71

Region (Region Host)	Hosts	IP Address
Back to Network [page 354]		
Back to Content [page 351]		

Region (Region Host)	Hosts	IP Address
----------------------	-------	------------

ABAP Environment

Note

To enable the full scenario for the SAP BTP ABAP environment, the DNS names below are needed *in addition* to the IPs of the Connectivity service mentioned above. If you can only configure allowlists for *IP addresses* in your firewall, please note that these addresses may change at any time and that you need to validate the values with a DNS service of your choice regularly.

For more information, see [Regions and API Endpoints for the ABAP Environment](#).

Europe (Frankfurt) - AWS	*.abap.eu10.hana.ondemand.com	<dynamic>
Europe (Frankfurt) EU Access - AWS	*.abap.eu11.hana.ondemand.com	<dynamic>
Europe (Netherlands) - Azure	*.abap.eu20.hana.ondemand.com	<dynamic>
US East (VA) - AWS	*.abap.us10.hana.ondemand.com	<dynamic>
US East (VA) - Azure	*.abap.us21.hana.ondemand.com	<dynamic>
US West (WA) - Azure	*.abap.us20.hana.ondemand.com	<dynamic>
Brazil (São Paulo) - AWS	*.abap.br10.hana.ondemand.com	<dynamic>
Japan (Tokyo) - AWS	*.abap.jp10.hana.ondemand.com	<dynamic>
Japan (Tokyo) - Azure	*.abap.jp20.hana.ondemand.com	<dynamic>
Australia (Sydney) - AWS	*.abap.ap10.hana.ondemand.com	<dynamic>
Australia (Sydney) - Azure	*.abap.ap20.hana.ondemand.com	<dynamic>
Asia Pacific (Singapore) - AWS	*.abap.ap11.hana.ondemand.com	<dynamic>
Asia Pacific (Seoul) - AWS	*.abap.ap12.hana.ondemand.com	<dynamic>
Singapore - Azure	*.abap.ap21.hana.ondemand.com	<dynamic>
Canada (Montreal) - AWS	*.abap.ca10.hana.ondemand.com	<dynamic>
Switzerland (Zurich) - Azure	*.abap.ch20.hana.ondemand.com	<dynamic>

Back to [Network \[page 354\]](#)

Back to [Content \[page 351\]](#)

Region (Region Host)	Hosts	IP Address
Neo Environment		
Europe (Rot)	connectivitynotification.hana.ondemand.com	155.56.210.83
(hana.ondemand.com)	connectivitycertsigning.hana.ondemand.com	155.56.210.43

Region (Region Host)	Hosts	IP Address
(eu1.hana.ondemand.com)	connectivitytunnel.hana.ondemand.com	155.56.210.84
Europe (Frankfurt)	connectivitynotification.eu2.hana.ondemand.com	157.133.206.143
(eu2.hana.ondemand.com)	connectivitycertsigning.eu2.hana.ondemand.com	157.133.205.174
	connectivitytunnel.eu2.hana.ondemand.com	157.133.205.233
Europe (Amsterdam)	connectivitynotification.eu3.hana.ondemand.com	130.214.87.72
(eu3.hana.ondemand.com)	connectivitycertsigning.eu3.hana.ondemand.com	130.214.87.76
	connectivitytunnel.eu3.hana.ondemand.com	130.214.86.212
United States East (Ashburn)	connectivitynotification.us1.hana.ondemand.com	130.214.181.204
(us1.hana.ondemand.com)	connectivitycertsigning.us1.hana.ondemand.com	130.214.181.48
	connectivitytunnel.us1.hana.ondemand.com	130.214.181.134
United States West (Chandler)	connectivitynotification.us2.hana.ondemand.com	130.214.129.127
(us2.hana.ondemand.com)	connectivitycertsigning.us2.hana.ondemand.com	130.214.129.140
	connectivitytunnel.us2.hana.ondemand.com	130.214.129.88
United States East (Sterling)	connectivitynotification.us3.hana.ondemand.com	130.214.32.144
(us3.hana.ondemand.com)	connectivitycertsigning.us3.hana.ondemand.com	130.214.33.92
	connectivitytunnel.us3.hana.ondemand.com	130.214.33.119
US States West (Colorado Springs)	connectivitynotification.us4.hana.ondemand.com	130.214.183.46
(us4.hana.ondemand.com)	connectivitycertsigning.us4.hana.ondemand.com	130.214.183.14
	connectivitytunnel.us4.hana.ondemand.com	130.214.183.103
Australia (Sydney)	connectivitynotification.ap1.hana.ondemand.com	157.133.97.47
(ap1.hana.ondemand.com)	connectivitycertsigning.ap1.hana.ondemand.com	157.133.97.27
	connectivitytunnel.ap1.hana.ondemand.com	157.133.97.46
China (Shanghai)	connectivitynotification.cn1.platform.sapcloud.cn	121.91.109.81
(cn1.platform.sapcloud.cn)	connectivitycertsigning.cn1.platform.sapcloud.cn	121.91.109.77
	connectivitytunnel.cn1.platform.sapcloud.cn	121.91.109.82
Japan (Tokyo)	connectivitynotification.jp1.hana.ondemand.com	130.214.112.168

Region (Region Host)	Hosts	IP Address
(jpl.hana.ondemand.com)	connectivitycertsigning.jp1.hana.ondemand.com	130.214.112.212
	connectivitytunnel.jp1.hana.ondemand.com	130.214.112.164
Canada (Toronto) (ca1.hana.ondemand.com)	connectivitynotification.ca1.hana.ondemand.com	130.214.174.201
	connectivitycertsigning.ca1.hana.ondemand.com	130.214.174.220
	connectivitytunnel.ca1.hana.ondemand.com	130.214.174.236
Brazil (São Paulo) (br1.hana.ondemand.com)	connectivitynotification.br1.hana.ondemand.com	130.214.96.193
	connectivitycertsigning.br1.hana.ondemand.com	130.214.96.195
	connectivitytunnel.br1.hana.ondemand.com	130.214.96.173
UAE (Dubai) (ae1.hana.ondemand.com)	connectivitynotification.ae1.hana.ondemand.com	130.214.80.206
	connectivitycertsigning.ae1.hana.ondemand.com	130.214.80.208
	connectivitytunnel.ae1.hana.ondemand.com	130.214.80.182
KSA (Riyadh) (sa1.hana.ondemand.com)	connectivitynotification.sa1.hana.ondemand.com	130.214.209.241
	connectivitycertsigning.sa1.hana.ondemand.com	130.214.209.207
	connectivitytunnel.sa1.hana.ondemand.com	130.214.209.191

Back to [Network \[page 354\]](#)

Back to [Content \[page 351\]](#)

Region (Region Host)	Hosts	IP Address
Trial (Cloud Foundry Environment)		
<div style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 10px;"> <p>Note</p> <p>In the Cloud Foundry environment, IPs are controlled by the respective IaaS provider (AWS, Azure, or Google Cloud). IPs may change due to network updates on the provider side. Any planned changes will be announced several weeks before they take effect. See also Regions.</p> </div>		
Europe (Frankfurt) - AWS (cf.eu10.hana.ondemand.com)	connectivitynotification.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223
	connectivitycertsigning.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223

Region (Region Host)	Hosts	IP Address
	connectivitytunnel.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223
United States East (VA) - AWS (cf.us10.hana.ondemand.com)	connectivitynotification.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211
	connectivitycertsigning.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211
	connectivitytunnel.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211
Singapore - Azure (cf.ap21.hana.ondemand.com)	connectivitynotification.cf.ap21.hana.ondemand.com	20.184.61.122
	connectivitycertsigning.cf.ap21.hana.ondemand.com	20.184.61.122
	connectivitytunnel.cf.ap21.hana.ondemand.com	20.184.61.122

[Back to Network \[page 354\]](#)

[Back to Content \[page 351\]](#)

Note

If you install the Cloud Connector in a network segment that is isolated from the backend systems, make sure the exposed hosts and ports are still reachable and open them in the firewall that protects them:

- for HTTP, the ports you chose for the HTTP/S server.
- for LDAP, the port of the LDAP server.
- for RFC, it depends on whether you use an SAProuter or not and whether load balancing is used:
 - if you use an SAProuter, it is typically configured as visible in the network of the Cloud Connector and the corresponding *routtab* is exposing all the systems that should be used.
 - without SAProuter, you must open the application server hosts and the corresponding gateway ports (33##, 48##). When using load balancing for the connection, you must also open the message server host and port.

See also [Network Zones \[page 368\]](#).

Note

For more information about the used ABAP server ports, see: [Ports of SAP NetWeaver Application Server ABAP](#).

For more information about additional IP addresses for SAP Business Application Studio, see [SAP Business Application Studio Inbound IP Addresses](#).

[Back to Network \[page 354\]](#)

[Back to Content \[page 351\]](#)

Related Information

[Installation on Microsoft Windows OS \[page 375\]](#)

[Installation on Linux OS \[page 378\]](#)

[Installation on Apple macOS \[page 381\]](#)

[Recommendations for Secure Setup \[page 383\]](#)

[Sizing Recommendations \[page 369\]](#)

1.2.1.1.1 System Requirements

Additional system requirements for installing and running the Cloud Connector.

Supported Browsers

The browsers you can use for the Cloud Connector Administration UI are the same as those currently supported by SAPUI5. See: [Browser and Platform Support](#).

Minimum Disk Space for Download and Installation

The minimum free disk space required to download and install a new Cloud Connector server is as follows:

- Size of downloaded Cloud Connector installation file (ZIP, TAR, MSI files): 50 MB
- Newly installed Cloud Connector server: 70 MB
- Total: 120 MB as a minimum

Additional Disk Space for Log and Configuration Files

The Cloud Connector writes configuration files, audit log files and trace files at runtime. We recommend that you reserve between 1 and 20 GB of disk space for those files.

Trace and log files are written to `<scd_dir>/log/` within the Cloud Connector `root` directory.

The `ljs_trace.log` file contains traces in general, communication payload traces are stored in `traffic_trace_*.trc`. These files may be used by SAP Support to analyze potential issues. The default trace level is `Information`, where the amount of written data is generally only a few KB per day. You can turn off these traces to save disk space. However, we recommend that you don't turn off this trace completely, but that you leave it at the default settings, to allow root cause analysis if an issue occurs. If you set the trace

level to `ALL`, the amount of data can easily reach the range of several GB per day. Use trace level `ALL` only to analyze a specific issue. Payload trace, however, should normally be turned off, and used only for analysis by SAP Support.

Note

Regularly back up or delete written trace files to clean up the used disk space.

Audit log files are written to `/log/audit/<subaccount-name>/audit-log_<subaccount-name>_<date>.csv` within the Cloud Connector root directory. By default, only security-related events are written in the audit log. You can change the audit log level using the administration UI, as described in [Manage Audit Logs \[page 690\]](#).

To be compliant with the regulatory requirements of your organization and the regional laws, the audit log files must be persisted for a certain period of time for traceability purposes. Therefore, we recommend that you back up the audit log files regularly from the Cloud Connector file system and keep the backup for the length of time required.

Related Information

[Prerequisites \[page 351\]](#)

1.2.1.1.2 Network Zones

Choose a network zone for your Cloud Connector installation.

A customer network is usually divided into multiple network zones or subnetworks according to the security level of the contained components. For example, the DMZ that contains and exposes the external-facing services of an organization to an untrusted network, usually the Internet, and there are one or more other network zones which contain the components and services provided in the company's intranet.

You can generally choose the network zone in which to set up the Cloud Connector:

- Internet access to the SAP BTP region host, either directly or via HTTPS proxy.
- Direct access to the internal systems it provides access to, which means that there is transparent connectivity between the Cloud Connector and the internal system.

The Cloud Connector can be set up either in the DMZ and operated centrally by the IT department, or set up in the intranet and operated by the appropriate line of business.

Note

The internal network must allow access to the required ports; the specific configuration depends on the firewall software used.

The default ports are **80** for HTTP and **443** for HTTPS. For RFC communication, you need to open a gateway port (default: **33+<instance number>**) and an arbitrary message server port. For a connection to a HANA Database (on SAP BTP) via JDBC, you need to open an arbitrary *outbound* port in your network. Mail (SMTP) communication is not supported.

1.2.1.2 Sizing Recommendations

When installing a Cloud Connector, the first thing you need to decide is the sizing of the installation.

This section gives some basic guidance what to consider for this decision. The provided information includes the shadow instance, which should always be added in productive setups. See also [Install a Failover Instance for High Availability \[page 652\]](#).

Note

The following recommendations are based on current experiences. However, they are only a rule of thumb since the actual performance strongly depends on the specific environment. The overall performance of a Cloud Connector is impacted by many factors (number of hosted subaccounts, bandwidth, latency to the attached regions, network routers in the corporate network, used JVM, and others).

Restrictions

The sizing data refer to a single Cloud Connector installation.

Note

Up until now, you cannot perform horizontal scaling directly. However, you can distribute the load statically by operating multiple Cloud Connector installations with different location IDs for all involved subaccounts. In this scenario, you can use multiple destinations with virtually the same configuration, except for the location ID. See also [Managing Subaccounts \[page 401\]](#), step 4. Alternatively, each of the Cloud Connector instances can host its own list of subaccounts without any overlap in the respective lists. Thus, you can handle more load, if a single installation risks to be overloaded.

Related Information

[Hardware Setup \[page 369\]](#)

[Configuration Setup \[page 373\]](#)

1.2.1.2.1 Hardware Setup

How to choose the right sizing for your Cloud Connector installation.

Regarding the hardware, we recommend that you use different setups for master and shadow. One dedicated machine should be used for the master, another one for the shadow. Usually, a shadow instance takes over the

master role only temporarily. During most of its lifetime, in the shadow state, it needs less resources compared to the master.

If the master instance is available again after a downtime, we recommend that you switch back to the actual master.

Note

The sizing recommendations refer to the overall load across all subaccounts that are connected via the Cloud Connector. This means that you need to accumulate the expected load of all subaccounts and should not only calculate separately per subaccount (taking the one with the highest load as basis).

Related Information

[Sizing for the Master Instance \[page 370\]](#)

[Sizing for the Shadow Instance \[page 372\]](#)

1.2.1.2.1.1 Sizing for the Master Instance

Learn more about the basic criteria for the sizing of your Cloud Connector master instance.

For the master setup, keep in mind the expected load for communication between the SAP BTP and on-premise systems. The setups listed below differ in a mostly qualitative manner, without hard limits for each of them.

Note

The mentioned sizes are considered as minimal configuration, larger ones are always ok. In general, the more applications, application instances, and subaccounts are connected, the more competition will exist for the limited resources on the machine.

Installation Size (S, M, L)	CPU (x86_64)	Machine Memory / Heap / Direct Memory	Disk Space
S: The expected load is small (up to 1 million requests per day, request concurrency and size is in average low) and only a few subaccounts with some applications are connected. In addition, only a few service channels are used, only small data amount is replicated to cloud systems. Setup can be a done in a virtual or physical machine.	2 cores 2.6 GHz	4GB RAM / 1GB / 2GB	10GB
M: The expected load is medium (up to 10 million requests per day, request concurrency and size is in average medium) and multiple subaccounts with multiple applications are connected. In addition, many service channels are used, and a medium data amount is replicated to cloud systems. We recommend that you do the setup in a virtual or physical machine. A virtual machine should be on a host without overcommitted resources.	4 cores 3.0 Ghz	16GB RAM / 4GB / 8GB	20GB

Installation Size (S, M, L)	CPU (x86_64)	Machine Memory / Heap / Direct Memory	Disk Space
L:	8 cores 3.0 Ghz	32GB RAM / 8GB / 16GB	40GB

The expected load is large (**more than 10 million requests** per day, request concurrency and size is in average medium or high) and multiple subaccounts with multiple applications are connected.

In addition, many service channels are used, and a **large data amount is replicated** to cloud systems concurrently.

We recommend that you do the setup in a virtual or physical machine. A virtual machine should be on a host without overcommitted resources.

Particularly the **heap size** is critical. If you size it too low for the load passing the Cloud Connector, at some point the Java Virtual Machine will execute full GCs (garbage collections) more frequently, blocking the processing of the Cloud Connector completely for multiple seconds, which massively slows down overall performance. If you experience such situations regularly, you should increase the heap size in the Cloud Connector UI (choose ► [Configuration](#) ► [Advanced](#) ► [JVM](#) ►). See also [Configure the Java VM \[page 622\]](#).

ⓘ Note

You should use the same value for both `<initial heap size>` and `<maximum heap size>`.

1.2.1.2.1.2 Sizing for the Shadow Instance

Learn more about the basic criteria for the sizing of your Cloud Connector shadow instance (high availability mode).

The shadow installation is typically not used in standard situations and hence does not need the same sizing, assuming that the time span in which it takes over the master role is limited.

ⓘ Note

The shadow only acts as master, for example, during an upgrade or when an abnormal situation occurs on the master machine, and either the Cloud Connector or the full machine on OS level needs to be restarted.

While being in the shadow state, the resource consumption is very low, especially in productive environments, where typically only few configuration changes are required. Therefore, the machine sizing can usually be smaller than the one for the master. However, if you want to mitigate the risk of a longer outage of the master machine, you should increase the sizing of the shadow up to the master size:

Master	Shadow
S	S installation as virtual machine
M	S installation (with double memory) as virtual or physical machine M installation as virtual or physical machine for risk mitigation
L	M installation as virtual or physical machine L installation as virtual or physical machine for risk mitigation

1.2.1.2.2 Configuration Setup

Choose the right connection configuration options to improve the performance of the Cloud Connector.

This section provides detailed information how you can adjust the configuration to improve overall performance. This is typically relevant for an M or L installation (see [Hardware Setup \[page 369\]](#)). For S installations, the default configuration will probably be sufficient to handle the traffic.

To change the connection parameters, proceed as follows:

- You can configure the number of physical connections through the Cloud Connector UI. See also [Configure Advanced Connectivity \[page 620\]](#).
- In versions prior to 2.11, you have to modify the configuration files with an editor and restart the Cloud Connector to activate the changes.

In general, the Cloud Connector tunnel is multiplexing multiple virtual connections over a single physical connection. Thus, a single connection can handle a considerable amount of traffic. However, increasing the maximum number of physical connections allows you to make use of the full available bandwidth and to minimize latency effects.

If the bandwidth limit of your network is reached, adding additional connections doesn't increase the throughput, but will only consume more resources.

Note

Different network access parameters may impact and limit your configuration options: if the access to an external network is a 1 MB line with an added latency of 50 ms, you will not be able to achieve the same data volumes like with a 10 GB line with an added latency of < 1 ms. However, even if the line is good, for example 10 GB, but with an added latency of 100 ms, the performance might still be bad.

Optimal configuration strongly depends on your actual scenarios. A good approach is to try out different settings, if the current performance does not meet your expectations.

Related Information

[On-Demand To On-Premise Connections \[page 374\]](#)

[On-Premise To On-Demand Connections \(Service Channels\) \[page 375\]](#)

1.2.1.2.2.1 On-Demand To On-Premise Connections

Configure the physical connections for on-demand to on-premise calls in the Cloud Connector.

Adjusting the number of physical connections for this direction is possible both globally in the Cloud Connector UI (► [Configuration](#) ► [Advanced](#) ►), and for individual communication partners on cloud side (► [On-Demand To On-Premise](#) ► [Applications](#) ►).

Connections are established for each defined and connected subaccount. The current number of opened connections is visible in the Cloud Connector UI via ► [<Subaccount>](#) ► [Cloud Connections](#) ►.

The global default is 1 physical connection per connected subaccount. This value is used across all subaccounts hosted by the Cloud Connector instance and applies for all communication partners.

In general, the default should be sufficient for applications with low traffic. If you expect medium traffic for most applications, it may be useful to set the default value to 2.

📘 Note

An exact traffic forecast is difficult to achieve. It requires a deep understanding of the use case and of the possible future load generated by different applications. For this reason, we recommend that you focus on subsequent configuration adjustments, using the Cloud Connector monitoring tools to recognize bottlenecks in time, and adjust Cloud Connector configuration accordingly.

Tunnel Worker Threads

In addition to the number of connections, you can configure the number of [<Tunnel Worker Threads>](#). This value should be at least equal to the maximum of all individual application tunnel connections in all subaccounts, to have at least 1 thread available for each connection that can process incoming requests and outgoing responses.

Protocol Processor Worker Threads

The value for <Protocol Processor Worker Threads> is mainly relevant if RFC is used as protocol. Since its communication model towards the ABAP system is a blocking one, each thread can handle only one call at a time and cannot be shared. Hence, you should provide 1 thread per 5 concurrent RFC requests.

Note

The longer the RFC execution time in the backend, the more threads you should provide. Threads can be reused only after the response of a call was returned to SAP BTP.

1.2.1.2.2 On-Premise To On-Demand Connections (Service Channels)

Configure the number of physical connections for a Cloud Connector service channel.

Service channels let you configure the number of physical connections to the communication partner on cloud side, see [Using Service Channels \[page 608\]](#). The default is 1. This value is used as well in versions prior to Cloud Connector 2.11, which did not offer a configuration option for each service channel. You should define the number of connections depending on the expected number of clients and, with lower priority, depending on the size of the exchanged messages.

If there is only a single RFC client for an S/4HANA Cloud channel or only a single HANA client for a HANA DB on SAP BTP side, increasing the number doesn't help, as each virtual connection is assigned to one physical connection. The following simple rule lets you to define the required number of connections per service channel:

- Per 10 concurrent clients, use one physical connection.
- If the transferred net data size is larger than 500k per request, make sure to add an additional connection per 2 of such clients.

Example

For a HANA system in the SAP BTP, data is replicated using 18 concurrent clients in the on-premise network. In average, about 5 of those clients are regularly sending 600k. For the number of clients, you should use 2 physical connections, for the 5 clients sending larger amounts add an additional 3, which sums up to 5 connections.

1.2.1.3 Installation on Microsoft Windows OS

Installing the Cloud Connector on a Microsoft Windows operating system.

Context

You can choose between a simple `portable` variant of the Cloud Connector and the MSI-based `installer`. The `installer` is the generally recommended version that you can use for both developer and productive scenarios. It lets you, for example, register the Cloud Connector as a Windows service and this way automatically start it after machine reboot.

→ Tip

If you are a developer, you might want to use the `portable` variant as you can run the Cloud Connector after a simple unzip (archive extraction). You might want to use it also if you cannot perform a full installation due to lack of permissions, or if you want to use multiple versions of the Cloud Connector simultaneously on the same machine.

Prerequisites

- You have one of the supported 64-bit operating systems. For more information, see [Product Availability Matrix \[page 353\]](#).
- You have downloaded either the `portable` variant as ZIP archive for Windows, or the MSI `installer` from the [SAP Development Tools for Eclipse](#) page.
- You must install Microsoft Visual Studio C++ 2013 *and* (for the latest SAP JVM versions) Microsoft Visual Studio C++ 2019 runtime libraries (download `vc_redist_x64.exe` for each version). For more information, see [Visual C++ Redistributable Packages for Visual Studio 2013](#) and [Microsoft Visual C++ Redistributable latest supported downloads](#).

ⓘ Note

Even with the more recent version 2019, you still must install the Microsoft Visual Studio C++ 2013 libraries, since they are needed for the Cloud Connector.

- A supported Java version must be installed. For more information, see [JDKs \[page 352\]](#). If you want to use SAP JVM, you can download it from the [SAP Development Tools for Eclipse](#) page.
- When using the `portable` variant, the environment variable `<JAVA_HOME>` must be set to the Java installation directory, so that the `bin` subdirectory can be found. Alternatively, you can add the relevant `bin` subdirectory to the `<PATH>` variable.

Portable Scenario

1. Extract the `<sapcc-<version>-windows-x64.zip>` ZIP file to an arbitrary directory on your local file system.
2. Set the environment variable `JAVA_HOME` to the installation directory of the JDK that you want to use to run the Cloud Connector. Alternatively, you can add the `bin` subdirectory of the JDK installation directory to the `PATH` environment variable.

3. Go to the Cloud Connector installation directory and start it using the `go.bat` batch file.
4. Continue with the **Next Steps** section.

Note

The Cloud Connector is not started as a service when using the portable variant, and hence will not automatically start after a reboot of your system. Also, the portable version does not support the automatic upgrade procedure. `sapcc-<version>-windows-x64.msi`

Installer Scenario

1. Start the `<>sapcc-` installer by double-clicking it.
2. The installer informs you that you are now guided through the installation process. Choose **Next>**.
3. Navigate to the desired installation directory for your Cloud Connector and choose **Next>**. When doing the installation in the context of an upgrade, make sure you choose the previous installation directory again.
4. You can choose the port on which the administration UI is reachable. Either leave the default **8443** or choose a different port if needed. Then, choose **Next>**.
5. Select the JDK to be used for running the Cloud Connector. The installer displays a list of all usable JDKs that are installed on your machine. If the needed JDK is not listed in the drop-down box (for example, if it's an SAP JVM that is not registered in the Windows registry upon installation), you can browse to its installation directory and select it. We recommend that you use an up-to-date Java **8** installation to run the Cloud Connector.
6. Decide whether the Cloud Connector should be started immediately after finishing the setup. Then, choose **Next>**.
7. To start the installation, press the **Next>** button again.
8. After successful installation, choose **Close**.
9. Continue with the **Next Steps** section.

Note

The Cloud Connector is started as a Windows service in the productive use case. Therefore, installation requires administration permissions. After installation, manage this service under **Control Panel > Administrative Tools > Services**. The service name is `Cloud Connector` (formerly named `Cloud Connector 2.0`). Make sure the service is executed with a user that has limited privileges. Typically, privileges allowed for service users are defined by your company policy. Adjust the folder and file permissions to be manageable by only this user and system administrators.

On Windows, the file `scc_service.log` is created and used by the Microsoft MSI installer (during Cloud Connector installation), and by the `scchost.exe` executable, which registers and runs the Windows service if you install the Cloud Connector as a Windows background job.

This log file is only needed if a problem occurs during Cloud Connector installation, or during creation and start of the Windows service, in which the Cloud Connector is running. You can find the file in the `log` folder of your Cloud Connector installation directory.

Starting the Cloud Connector

After installation, the Cloud Connector is registered as a Windows service that is configured to be started automatically after a system reboot. You can start and stop the service via shortcuts on the desktop ("Start Cloud Connector" and "Stop Cloud Connector"), or by using the `Windows Services` manager and look for the service **SAP Cloud Connector**.

Access the Cloud Connector administration UI at <https://localhost:<port>>, where the default port is **8443** (but this port might have been modified during the installation).

Next Steps

1. Open a browser and enter: <https://<hostname>:8443>. `<hostname>` is the host name of the machine on which you have installed the Cloud Connector. If you access the Cloud Connector locally from the same machine, you can simply enter `localhost`.
2. Continue with the initial configuration of the Cloud Connector, see [Initial Configuration \[page 388\]](#).

Related Information

[Recommendations for Secure Setup \[page 383\]](#)

1.2.1.4 Installation on Linux OS

Installing the Cloud Connector on a Linux operating system.

Context

You can choose between a simple `portable` variant of the Cloud Connector and the RPM-based `installer`. The `installer` is the generally recommended version that you can use for both the developer and the productive scenario. It registers, for example, the Cloud Connector as a daemon service and this way automatically starts it after machine reboot.

→ Tip

If you are a developer, you might want to use the `portable` variant as you can run the Cloud Connector after a simple `tar -xzf` execution. You also might want to use it if you cannot perform a full installation due to missing permissions for the operating system, or if you want to use multiple versions of the Cloud Connector simultaneously on the same machine.

Prerequisites

- You have one of the supported 64-bit operating systems. For more information, see [Product Availability Matrix \[page 353\]](#).
- The supported platforms are `x64` and `ppc64le`, represented below by the variable `<platform>`. Variable `<arch>` is `x86_64` or `ppc64le` respectively.
- You have downloaded either the `portable` variant as `tar.gz` archive for Linux or the RPM installer contained in the ZIP for Linux, from [SAP Development Tools for Eclipse](#).
- A supported Java version must be installed. For more information, see [JDKs \[page 352\]](#). If you want to use SAP JVM, you can download it from the [SAP Development Tools for Eclipse](#) page. Use the following command to install it:

```
rpm -i sapjvm-<version>-linux-<platform>.rpm
```

If you want to check the JVM version installed on your system, use the following command:

```
rpm -qa | grep jvm
```

When installing it using the RPM package, the Cloud Connector will detect it and use it for its runtime.

- When using the `tar.gz` archive, the environment variable `<JAVA_HOME>` must be set to the Java installation directory, so that the `bin` subdirectory can be found. Alternatively, you can add the Java installation's `bin` subdirectory to the `<PATH>` variable.

Portable Scenario

1. Extract the `tar.gz` file to an arbitrary directory on your local file system using the following command:

```
tar -xzf sapcc-<version>-linux-<platform>.tar.gz
```

Note

If you use the parameter "o", the extracted files are assigned to the user ID and the group ID of the user who has unpacked the archive. This is the default behavior for users other than the `root` user.

2. Go to this directory and start the Cloud Connector using the `go.sh` script.
3. Continue with the [Next Steps](#) section.

Note

In this case, the Cloud Connector is not started as a daemon, and therefore will not automatically start after a reboot of your system. Also, the `portable` version does not support the automatic upgrade procedure.

Installer Scenario

1. Extract the `sapcc-<version>-linux-<platform>.zip` archive to an arbitrary directory by using the following the command:

```
unzip sapcc-<version>-linux-<platform>.zip
```

2. Go to this directory and install the extracted RPM using the following command. You can perform this step only as a root user.

```
rpm -i com.sap.scc-ui-<version>.<arch>.rpm
```

3. Continue with the **Next Steps** section.

In the productive case, the Cloud Connector is started as a daemon. If you need to manage the daemon process, execute:

```
System V init distributions: service scc_daemon stop|restart|start|status
systemd distributions: systemctl stop|restart|start|status scc_daemon
```

⚠ Caution

When adjusting the Cloud Connector installation (for example, restoring a backup), make sure the RPM package management is synchronized with such changes. If you simply replace files that do not fit to the information stored in the package management, lifecycle operations (such as upgrade or uninstallation) might fail with errors. Also, the Cloud Connector might get into unrecoverable state.

Example: After a file system restore, the system files represent Cloud Connector 2.3.0 but the RPM package management "believes" that version 2.4.3 is installed. In this case, commands like `rpm -U` and `rpm -e` do not work as expected. Furthermore, avoid using the `--force` parameter as it may lead to an unpredictable state with two versions being installed concurrently, which is not supported.

Extending the Daemon

When using SNC for encrypting RFC communication, it might be required to provide some settings, for example, environment variables that must be visible for the Cloud Connector process. To achieve this, you must store a file named `scc_daemon_extension.sh` in the installation directory of the Cloud Connector (`/opt/sap/scc`), containing all commands needed for initialization without a shebang.

Example (SAP Cryptographic Library requires SECUDIR to be set):

📄 Sample Code

```
export SECUDIR=/path/to/psefile
```

To activate it, you must reinstall the daemon.

📌 Note

Make sure `JAVA_HOME` is set to the JVM used (in the shell where the command is executed).

Execute the following command:

```
System V init distributions: /opt/sap/scc/daemon.sh reinstall
```

```
systemd distributions: /opt/sap/scc/daemon.sh reinstallSystemd
```

The daemon extension will survive Cloud Connector version updates.

Starting the Cloud Connector

After installation via RPM manager, the Cloud Connector process is started automatically and registered as a daemon process, which ensures the automatic restart of the Cloud Connector after a system reboot.

To start, stop, or restart the process explicitly, open a command shell and use the following commands, which require `root` permissions:

```
System V init distributions: service scc_daemon start|stop|restart
systemd distributions: systemctl start|stop|restart scc_daemon
```

Next Steps

1. Open a browser and enter: <https://<hostname>:8443>. `<hostname>` is the host name of the machine on which you installed the Cloud Connector.
If you access the Cloud Connector locally from the same machine, you can simply enter `localhost`.
2. Continue with the initial configuration of the Cloud Connector, see [Initial Configuration \[page 388\]](#).

Related Information

[Recommendations for Secure Setup \[page 383\]](#)

1.2.1.5 Installation on Apple macOS

Installing the Cloud Connector on an Apple macOS operating system.

Prerequisites

Note

Apple macOS is not supported for productive scenarios. The developer version described below must not be used as productive version.

⚠ Caution

There are two different Cloud Connector portable versions available for running on Apple macOS with native support either for Apple M1/M2 CPUs based on the *aarch64* architecture, or with native support for INTEL x86 64-bit CPUs based on the *x64* architecture. Make sure you download and use the Cloud Connector version in combination with a JVM version, which both match your used hardware CPU architecture.

- You have one of the supported 64-bit operating systems. For more information, see [Product Availability Matrix \[page 353\]](#).
- The supported platforms are *aarch64* and *x64*, represented below by the variable `<platform>`.
- You have downloaded the `tar.gz` archive for the developer use case on Apple macOS from [SAP Development Tools for Eclipse](#).
- A supported Java version must be installed. For more information, see [JDKs \[page 352\]](#).
If you are running Apple macOS on an INTEL x86 64-bit CPU and you want to use SAP JVM, you can download it from the [SAP Development Tools for Eclipse](#) page.
- Environment variable `<JAVA_HOME>` must be set to the Java installation directory so that the `bin` subdirectory can be found. Alternatively, you can add the Java installation's `bin` subdirectory to the `<PATH>` variable.

Procedure

1. Extract the `tar.gz` file to an arbitrary directory on your local file system using the following command:

```
tar -xzf sapcc-<version>-macosx-<platform>.tar.gz
```

2. Go to this directory and start Cloud Connector using the `go.sh` script.
3. Continue with the [Next Steps](#) section.

📘 Note

The Cloud connector is not started as a daemon, and therefore will not automatically start after a reboot of your system. Also, the macOS version of the Cloud Connector does not support the automatic upgrade procedure.

Next Steps

1. Open a browser and enter: `https://<hostname>:8443`. `<hostname>` is the host name of the machine on which you installed the Cloud Connector.
If you access the Cloud Connector locally from the same machine, you can simply enter `localhost`.
2. Continue with the initial configuration of the Cloud Connector, see [Initial Configuration \[page 388\]](#).

Related Information

[Recommendations for Secure Setup \[page 383\]](#)

1.2.1.6 Recommendations for Secure Setup

For the Connectivity service and the Cloud Connector, you should apply the following guidelines to guarantee the highest level of security for these components.

Security Status

From the *Connector* menu, choose *Security Status* to access an overview showing potential security risks and the recommended actions.

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar contains a navigation menu with 'Security Status' highlighted. The main content area is titled 'Security Status' and features a red 'Risk' icon. Below this, there are two tables:

General Security Status			
Status	Area	Description	Actions
⚠	UI Certificate	Replace the default UI certificate with a certificate that uses the host name as its common name (CN)	>
⚠	Cipher Suites	Only TLS ciphers with SHA256 (or greater bit lengths) are deemed secure	>
⚠	Trust Store	Trust store is empty — no access restrictions	>
⚠	Authentication	Configure local LDAP for authentication of Cloud Connector users	>
✅	CPIC Trace	Trace is off	>
⚠	Service User	Set up service user specifically for this Cloud Connector	✎

Subaccount-Specific Security Status		
Display Name	Application White-List	Payload Trace
conn2	⚠ White-list is empty — all applications will be trusted	✅ Trace is off
conn3	⚠ White-list is empty — all applications will be trusted	✅ Trace is off

The *General Security Status* addresses security topics that are subaccount-independent.

- Choose any of the *Actions* icons in the corresponding line to navigate to the UI area that deals with that particular topic and view or edit details.

Note

Navigation is not possible for the last item in the list (*Service User*).

- The service user is specific to the Windows operating system (see [Installation on Microsoft Windows OS \[page 375\]](#) for details) and is only visible when running the Cloud Connector on Windows. It cannot be accessed or edited through the UI. If the service user was set up properly, choose *Edit* and check the corresponding checkbox.

The *Subaccount-Specific Security Status* lists security-related information for each and every subaccount.

Note

The security status only serves as a reminder to address security issues and shows if your installation complies with all recommended security settings.

UI Access

Upon installation, the Cloud Connector provides an initial user name and password for the administration UI, and forces the user (**Administrator**) to change the password. You must change the password immediately after installation.

The connector itself does not check the strength of the password. You should select a strong password that cannot be guessed easily.

Note

To enforce your company's password policy, we recommend that you configure the Administration UI to use an LDAP server for authorizing access to the UI.

The default user store is a local file store. It allows only one user, and only the *Administrator* role for this user. Using an LDAP server as user store lets you create various users to access the UI, and assign different roles to them. For more information on available roles, see [Use LDAP for User Administration \[page 634\]](#).

The Cloud Connector administration UI can be accessed remotely via HTTPS. The connector uses a standard X.509 self-signed certificate as SSL server certificate. You can exchange this certificate with a specific certificate that is trusted by your company. See [Exchange UI Certificates in the Administration UI \[page 629\]](#).

Note

Since browsers usually do not resolve *localhost* to the host name whereas the certificate usually is created under the host name, you might get a certificate warning. In this case, simply skip the warning message.

OS-Level Access

The Cloud Connector is a security-critical component that handles the external access to systems of an isolated network, comparable to a reverse proxy. We therefore recommend that you restrict the access to the operating system on which the Cloud Connector is installed to the minimal set of users who would administrate the Cloud Connector. This minimizes the risk of unauthorized users getting access to credentials, such as certificates stored in the secure storage of the Cloud Connector.

We also recommend that you use the machine to operate only the Cloud Connector and no other systems.

Administrator Privileges

To log on to the Cloud Connector administration UI, the **Administrator** user of the connector must not have an operating system (OS) user for the machine on which the connector is running. This allows the OS administrator to be distinguished from the Cloud Connector administrator. To make an initial connection between the connector and a particular SAP BTP subaccount, you need an SAP BTP user with the required permissions for the related subaccount. We recommend that you separate these roles/duties (that means, you have separate users for Cloud Connector administrator and SAP BTP).

Note

We recommend that only a small number of users are granted access to the machine as **root** users.

Hard Drive Encryption

Hard drive encryption for machines with a Cloud Connector installation ensures that the Cloud Connector configuration data cannot be read by unauthorized users, even if they obtain access to the hard drive.

Supported Protocols

Currently, the protocols HTTP, HTTPS, RFC, RFC with SNC, LDAP, LDAPS, TCP, and TCP over TLS are supported for connections between the SAP BTP and on-premise systems when the Cloud Connector and the Connectivity service are used. The whole route from the application virtual machine in the cloud to the Cloud Connector is always SSL-encrypted.

The route from the connector to the back-end system can be TLS-encrypted or SNC-encrypted. See [Configure Access Control \(HTTP\) \[page 457\]](#) and [Configure Access Control \(RFC\) \[page 465\]](#).

Audit Log on OS Level

We recommend that you turn on the audit log on operating system level to monitor the file operations.

Audit Log on Cloud Connector Level

The Cloud Connector audit log must remain switched on during the time it is used with productive systems. The default audit level is **SECURITY**. Set it to **ALL** if required by your company policy. The administrators who

are responsible for a running Cloud Connector must ensure that the audit log files are properly archived, to conform to the local regulations. You should switch on audit logging also in the connected back-end systems.

Encryption Ciphers

→ Tip

Enable all cipher suites that are compatible with the current UI certificate and are deemed secure as per your company's and SAP's guidelines. Adapt the cipher suites whenever the UI certificate is exchanged or the JVM is updated.

Initially, a default set of encryption ciphers is enabled for HTTPS connections to the administration UI. This default set is determined by the JVM. Some of these ciphers may not be compliant with the UI certificate, and some of them may not conform to your or SAP's security standards. They should therefore be excluded.

To enable or disable ciphers, choose *Configuration* from the main menu and go to tab *User Interface*, section *Cipher Suites*.

The screenshot shows the 'Configuration' page in the SAP BTP Administration UI. The 'Cipher Suites' section is highlighted with a red box. The table below shows the details of the cipher suites:

Status Quo	Status New	Security	Name	Actions
◇	□	✓	TLS_AES_128_GCM_SHA256	⊗
◇	□	✓	TLS_AES_256_GCM_SHA384	⊗
△		⚠	TLS_DHE_DSS_WITH_AES_128_CBC_SHA	⊗
△		⚠	TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	⊗
△	⊗	✓	TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	⊗
△		⚠	TLS_DHE_DSS_WITH_AES_256_CBC_SHA	⊗
△		⚠	TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	⊗
△		✓	TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	⊗
□		⚠	TLS_DHE_RSA_WITH_AES_128_CBC_SHA	⊗
□		⚠	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	⊗

The first column labeled **Status Quo** shows the current state of all available ciphers. The second column **Status New** shows the state the ciphers will have after a restart, if that state differs from the current one (that is, there is no entry in that column if the state remains the same after a restart).

Ciphers are either enabled or disabled, and they are either compatible or incompatible with the current UI certificate (that is, can potentially be used or not be used). Consult the tooltips of the (four types of) icons for details. The third column shows SAP's security assessment of the ciphers *as per the time of release*. Enable or disable individual ciphers using the button in the **Action** column. Enable or disable certain groups of ciphers using the appropriate table button. Consult the tooltips for details.

Note

We recommend that you enable all cipher suites that are compatible with the UI whenever you plan to switch to another JVM. You can comfortably do so by using the first button to the right of the filter buttons.

As the set of supported ciphers may differ, the selected ciphers may not be supported by the new JVM. In that case, the Cloud Connector will not start anymore, and you must fix the issue by manually adapting the file `conf/server.xml`. After a successful switch, you can adjust the list of eligible ciphers again.

Related Information

[Connectivity via Reverse Proxy \[page 867\]](#)

[Security \[page 703\]](#)

1.2.2 Configuration

Configure the Cloud Connector to make it operational for connections between your SAP BTP applications and on-premise systems.

Topic	Description
Initial Configuration [page 388]	After installing the Cloud Connector and starting the Cloud Connector daemon, you can log on and perform the required configuration to make your Cloud Connector operational.
Managing Subaccounts [page 401]	How to connect SAP BTP subaccounts to your Cloud Connector.
Authenticating Users against On-Premise Systems [page 419]	Basic authentication and principal propagation (user propagation) are the authentication types currently supported by the Cloud Connector.
Configure Access Control [page 456]	Configure access control or copy the complete access control settings from another subaccount on the same Cloud Connector.
Configuration REST APIs [page 482]	Configure a newly installed Cloud Connector (initial configuration, subaccounts, access control) using the configuration REST API.
Using Service Channels [page 608]	Service channels provide access from an external network to certain services on SAP BTP, which are not exposed to direct access from the Internet.
Configure Trust [page 616]	Set up an allowlist for trusted cloud applications and a trust store for on-premise systems in the Cloud Connector.

Topic	Description
Connect DB Tools to SAP HANA via Service Channels [page 609]	How to connect database, BI, or replication tools running in the on-premise network to a HANA database on SAP BTP using the service channels of the Cloud Connector.
Configure Domain Mappings for Cookies [page 618]	Map virtual and internal domains to ensure correct handling of cookies in client/server communication.
Configure Solution Management Integration [page 619]	Activate Solution Management reporting in the Cloud Connector.
Configure Advanced Connectivity [page 620]	Adapt connectivity settings that control the throughput and HTTP connectivity to on-premise systems.
Configure the Java VM [page 622]	Adapt the JVM settings that control memory management.
Configuration Backup [page 623]	Backup and restore your Cloud Connector configuration.
Configure Login Screen Information [page 625]	Add additional information to the login screen and configure its appearance.

1.2.2.1 Initial Configuration

After installing and starting the Cloud Connector, log on to the administration UI and perform the required configuration to make your Cloud Connector operational.

Tasks

[Prerequisites \[page 388\]](#)

[Log on to the Cloud Connector \[page 390\]](#)

[Initial Setup \[page 390\]](#)

[Set up Connection Parameters and HTTPS Proxy \[page 391\]](#)

[Establish Connections to SAP BTP \[page 395\]](#)

Prerequisites

- You have downloaded and installed the Cloud Connector, see [Installation \[page 349\]](#).
- You have assigned one of these roles/role collections to the subaccount user that you use for initial Cloud Connector setup, depending on the SAP BTP environment in which your subaccount is running:

Note

For the **Cloud Foundry** environment, you must know on which cloud management tools feature set (A or B) your account is running. For more information on feature sets, see [Cloud Management Tools — Feature Set Overview](#).

Environment	Required Roles/Role Collections	More Information
Cloud Foundry [feature set A]	<p>The user must be a <i>member of the global account</i> that the subaccount belongs to.</p> <p>Alternatively, you can assign the user as <i>Security Administrator</i>.</p>	<p>Add Members to Your Global Account</p> <p>Managing Security Administrators in Your Subaccount [Feature Set A]</p>
Cloud Foundry [feature set B]	<p>Assign at least one of these <i>default role collections</i> (all of them including the role <code>Cloud Connector Administrator</code>):</p> <ul style="list-style-type: none">• <code>Subaccount Administrator</code>• <code>Cloud Connector Administrator</code>• <code>Connectivity and Destination Administrator</code> <p>Alternatively, you can assign a <i>custom role collection</i> to the user that includes the role <code>Cloud Connector Administrator</code>.</p>	<p>Default Role Collections [Feature Set B] [page 15]</p> <p>Role Collections and Roles in Global Accounts, Directories, and Subaccounts [Feature Set B]</p>
Neo	<p>Assign at least one of these <i>default roles</i>:</p> <ul style="list-style-type: none">• <code>Cloud Connector Admin</code>• <code>Administrator</code> <p>Alternatively, you can assign a <i>custom role</i> to the user that includes the permission <code>manageSCCTunnels</code>.</p>	<p>Managing Member Authorizations in the Neo Environment</p>

After establishing the Cloud Connector connection, this user is not needed any more, since it serves only for initial connection setup. You may revoke the corresponding role assignment then and remove the user from the *Members* list (**Neo** environment), or from the *Users* list (**Cloud Foundry** environment).

Note

If the Cloud Connector is installed in an environment that is operated by SAP, SAP provides a user that you can add as member in your SAP BTP subaccount and assign the required role.

- We strongly recommend that you read and follow the steps described in [Recommendations for Secure Setup \[page 383\]](#). For operating the Cloud Connector securely, see also [Security Guidelines \[page 711\]](#).

Back to [Tasks \[page 388\]](#)

Log on to the Cloud Connector

To administer the Cloud Connector, you need a Web browser. To check the list of supported browsers, see [Prerequisites and Restrictions](#) → section *Browser Support*.

1. In a Web browser, enter: **https://<hostname>:<port>**
 - **<hostname>** refers to the machine on which the Cloud Connector is installed. If installed on your machine, you can simply enter **localhost**.
 - **<port>** is the Cloud Connector port specified during installation (the default port is **8443**).
2. On the logon screen, enter **Administrator / manage** (case sensitive) for **<User Name> / <Password>**.

Note

By default, the Cloud Connector includes a self-signed UI certificate. Browsers may show a security warning because they don't trust the issuer of this certificate. In this case, you can skip the warning message.

Back to [Tasks \[page 388\]](#)

Initial Setup

1. When you first log in, you must change the password before you continue, regardless of the installation type you have chosen.
2. Choose between master and shadow installation. Use *Master* if you are installing a single Cloud Connector instance or a main instance from a pair of Cloud Connector instances. See [Install a Failover Instance for High Availability \[page 652\]](#).

SAP Cloud Connector Administration Administrator

Initial Setup Save ?

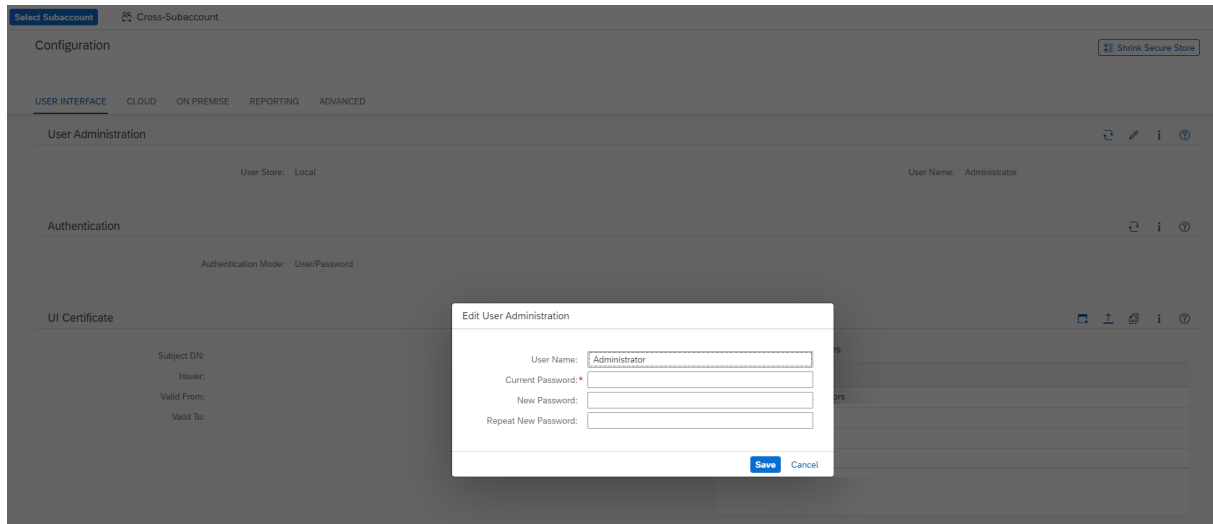
You are required to change your password before being permitted to continue

Mandatory Password Change	Choose Installation Type
Current Password: <input type="text"/>	<input checked="" type="radio"/> Master (Primary Installation)
New Password: <input type="text"/>	<input type="radio"/> Shadow (Backup Installation)
Repeat New Password: <input type="text"/>	Description: <input type="text"/>

3. (Optional): When configuring a master, you can provide a (free-text) *Description* for this Cloud Connector instance that helps you distinguish different Cloud Connectors. This information will also be shown in the *Cloud Connectors* view in the SAP BTP cockpit.

User Administration

To edit the password for the **Administrator** user, choose *Configuration* from the main menu, tab *User Interface*, section *User Administration*:



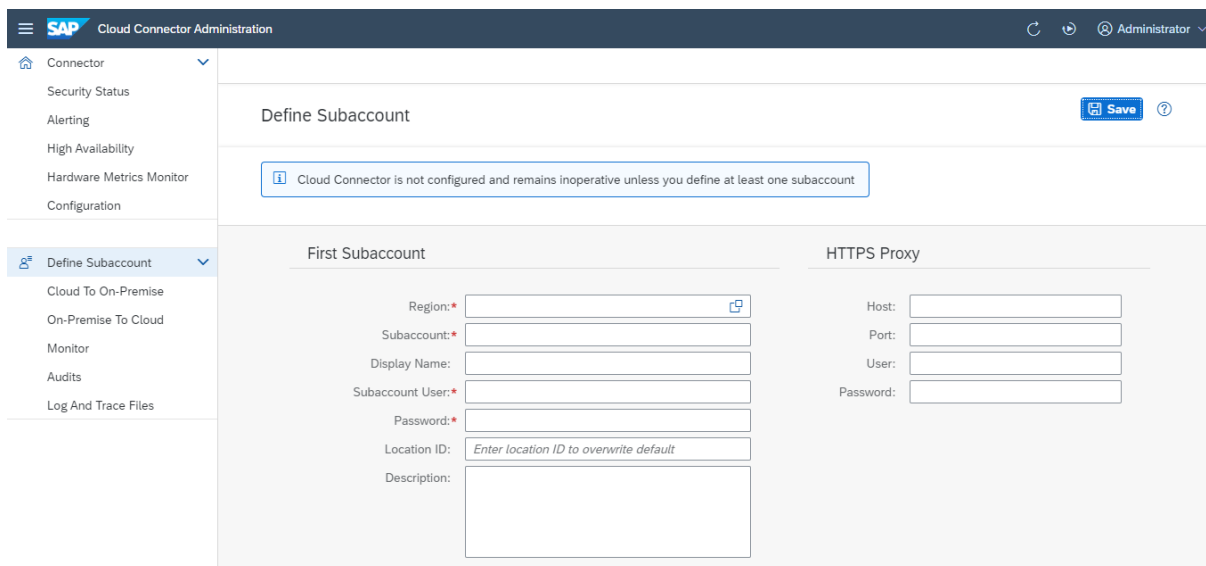
Note

User name and password cannot be changed at the same time. If you want to change the user name, you must enter only the current password in a first step. Do not enter values for **<New Password>** or **<Repeat New Password>** when changing the user name. To change the password in second step, enter the old password, the new one, and the repeated (new) password, but leave the user name unchanged.


Back to [Tasks \[page 388\]](#)

Set up Connection Parameters and HTTPS Proxy

When logging in for the first time, the following screen is displayed every time you choose an option from the main menu that requires a configured subaccount:



Note

If you want to skip the initial configuration, you can click the  icon in the upper right corner. You might need this in case of connectivity issues shown in your logs. You can add subaccounts later as described in [Managing Subaccounts \[page 401\]](#).

The Cloud Connector collects the following required information for your subaccount connection:

1. For **<Region>**, specify the SAP BTP region that should be used. You can choose it from the drop-down list, see [Regions](#).

Note

You can also configure a region yourself, if it is not part of the standard list. Either insert the region host manually, or create a custom region, as described in [Configure Custom Regions \[page 418\]](#).

2. For **<Subaccount>**, **<Subaccount User>** and **<Password>**, enter the values you obtained when you registered your subaccount on SAP BTP.

Note

For a subaccount in the **Cloud Foundry** environment, you must enter the subaccount ID as **<Subaccount>**, rather than its actual (technical) name. For information on getting the subaccount ID, see [Find Your Subaccount ID \(Cloud Foundry Environment\) \[page 417\]](#). As **<Subaccount User>** you must provide your **Login E-mail** instead of a user ID. The user must be a member of the global account the subaccount belongs to.

You can also add a new subaccount user with the role `Cloud_Connector_Admin` in the SAP BTP cockpit and use the new user and password.

Note

The Cloud Connector does not yet support *SAP Universal ID*. Please use your S-user or P-user credentials for the **<subaccount user>** and **<password>** fields instead.

For more information, see SAP note [3085908](#).

For the **Neo** environment, see [Add Members to Your Neo Subaccount](#).

For the **Cloud Foundry** environment, see [Add Org Members Using the Cockpit](#).

→ Tip

When using SAP Cloud Identity Services - Identity Authentication (IAS) as platform identity provider with two-factor authentication for your subaccount, you can simply append the required token to the regular password.

→ Tip

For a subaccount in the **Cloud Foundry** environment, the Cloud Connector supports the use of a custom identity provider (IDP) via single sign-on (SSO) passcode. For more information, see [Use a Custom IDP for Subaccount Configuration \[page 409\]](#).

- (Optional) You can define a `<Display Name>` that lets you easily recognize a specific subaccount in the UI compared to the technical subaccount name.
- (Optional) You can define a `<Location ID>` identifying the location of this Cloud Connector for a specific subaccount. The location ID is used as routing information and therefore you can connect multiple Cloud Connectors to a single subaccount. If you don't specify any value for `<Location ID>`, the default is used, which represents the behavior of previous Cloud Connector versions. 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.

ⓘ Note

Location IDs provided in older versions of the Cloud Connector are discarded during upgrade to ensure compatibility for existing scenarios.

- (Optional) Enter proxy host and port. Omit the proxy configuration unless your internal landscape is protected by a firewall that blocks any outgoing TCP traffic. In that case, you must specify an HTTPS proxy that the Cloud Connector can use to connect to SAP BTP. The Cloud Connector performs two operations for which it may need a proxy in the situation outlined above:
 - Downloading the correct connection configuration corresponding to your subaccount ID in SAP BTP.
 - Establishing the TLS tunnel connection from the Cloud Connector to your SAP BTP subaccount.

ⓘ Note

A proxy server is required to support TLS communication; a standard HTTP proxy will not suffice. Typically, you choose the same proxy settings as those being used by your standard Web browser.

ⓘ Note

Some proxy servers require credentials for authentication. In this case, you need to provide the relevant user/password information.

- (Optional) You can provide a `<Description>` (free-text) of the subaccount that is shown when choosing the *Details* icon in the *Actions* column of the *Subaccount Dashboard*. It lets you identify the particular Cloud Connector you use.

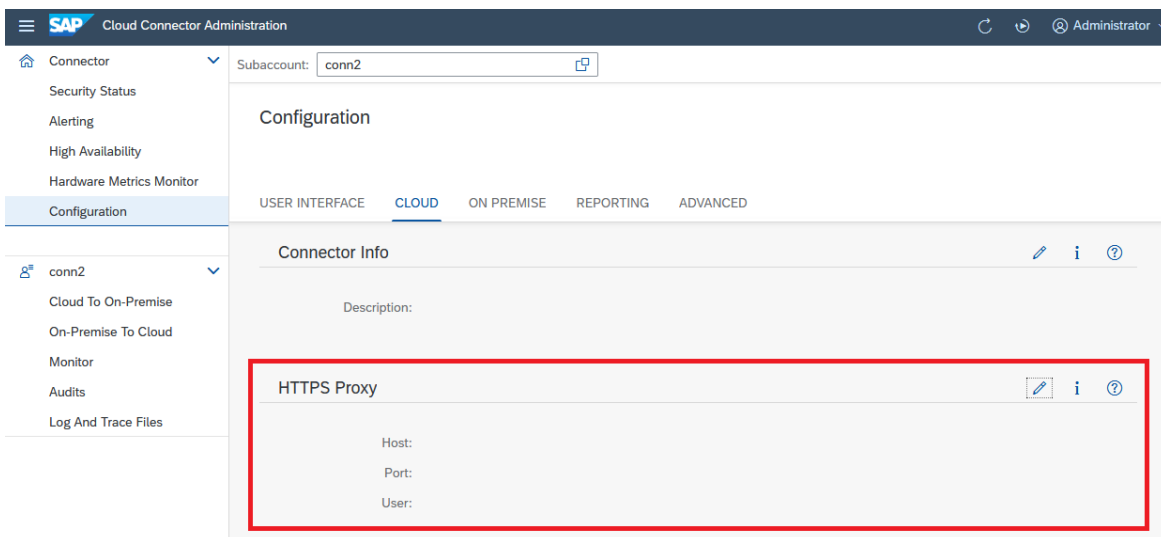
- 7. Choose [Save](#).

The Cloud Connector now starts a handshake with SAP BTP and attempts to establish a secure TLS tunnel to the server that hosts the subaccount in which your on-demand applications are running. However, no requests are yet allowed to pass from the cloud side to any of your internal backend systems. To allow your on-demand applications to access specific internal back-end systems, proceed with the access configuration described in the next section.

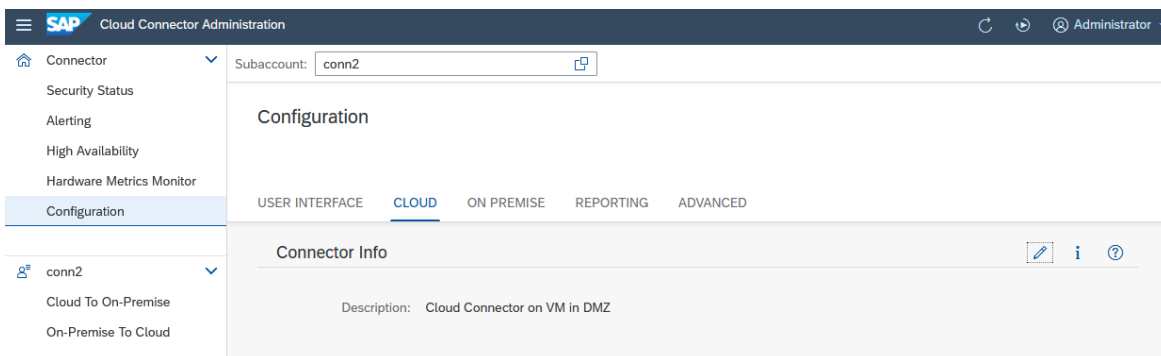
Note

The internal network must allow access to the port. Specific configuration for opening the respective port(s) depends on the firewall software used. The default ports are **80** for HTTP and **443** for HTTPS. For RFC communication, you must open a gateway port (default: **33+<instance number>**) and an arbitrary message server port. For a connection to a HANA Database (on SAP BTP) via JDBC, you must open an arbitrary *outbound* port in your network. Mail (SMTP) communication is not supported.

- If you later want to change your proxy settings (for example, because the company firewall rules have changed), choose [Configuration](#) from the main menu and go to the [Cloud](#) tab, section [HTTPS Proxy](#). Some proxy servers require credentials for authentication. In this case, you must provide the relevant user/password information.



- If you want to change the description for your Cloud Connector, choose [Configuration](#) from the main menu, go to the [Cloud](#) tab, section [Connector Info](#) and edit the description:

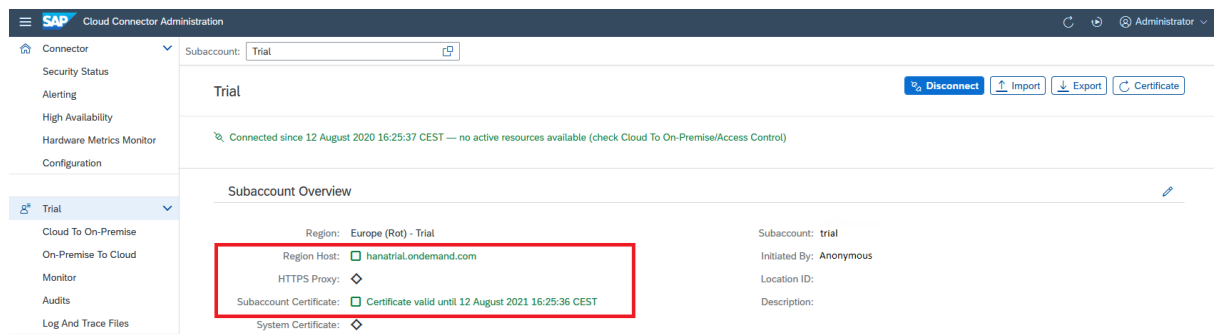


[Back to Tasks \[page 388\]](#)

Establish Connections to SAP BTP

As soon as the initial setup is complete, the tunnel to the cloud endpoint is open, but no requests are allowed to pass until you have performed the *Access Control* setup, see [Configure Access Control \[page 456\]](#).

To manually close (and reopen) the connection to SAP BTP, choose your subaccount from the main menu and select the *Disconnect* button (or the *Connect* button to reconnect to SAP BTP).



- The green icon next to *Region Host* indicates that it is valid and can be reached.
- If an *HTTPS Proxy* is configured, its availability is shown the same way. In the screenshot, the grey diamond icon next to *HTTPS Proxy* indicates that connectivity is possible without proxy configuration.

In case of a timeout or a connectivity issue, these icons are yellow (warning) or red (error), and a tooltip shows the cause of the problem. *Initiated By* refers to the user that has originally established the tunnel. During normal operations, this user is no longer needed. Instead, a certificate is used to open the connection to a subaccount.

- The status of the certificate is shown next to *Subaccount Certificate*. It is shown as valid (green icon), if the expiration date is still far in the future, and turns to yellow if expiration approaches according to your alert settings. It turns red as soon as it has expired. This is the latest point in time, when you should [Update the Certificate for Your Subaccount \[page 413\]](#).

Note

When connected, you can monitor the Cloud Connector also in the *Connectivity* section of the SAP BTP cockpit. There, you can track attributes like version, description and high availability set up. Every Cloud Connector configured for your subaccount automatically appears in the *Connectivity* section of the cockpit.

[Back to Tasks \[page 388\]](#)

Related Information

[Managing Subaccounts \[page 401\]](#)

[Initial Configuration \(HTTP\) \[page 396\]](#)

[Initial Configuration \(RFC\) \[page 398\]](#)

[Configuring the Cloud Connector for LDAP \[page 401\]](#)

[Managing Member Authorizations in the Neo Environment](#)

[Use a Custom IDP for Subaccount Configuration \[page 409\]](#)

1.2.2.1.1 Initial Configuration (HTTP)

Configure the Cloud Connector for HTTP communication.

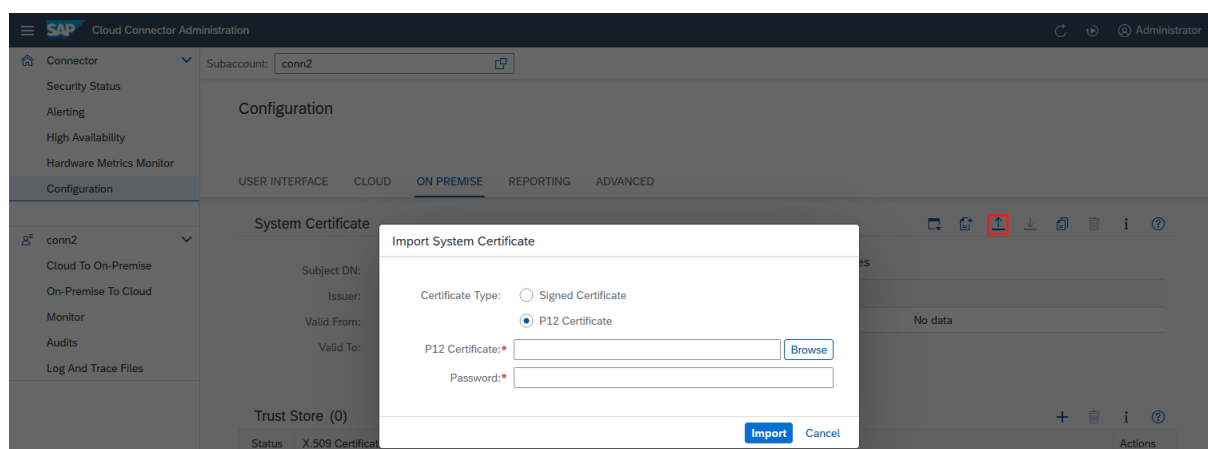
Install a System Certificate for Mutual Authentication

To set up a mutual authentication between the Cloud Connector and any backend system it connects to, you can import an X.509 client certificate into the Cloud Connector. The Cloud Connector then uses the so-called *system certificate* for all HTTPS requests to backends that request or require a client certificate. The certificate authority (CA) that signed the Cloud Connector's system certificate must be trusted by all backend systems to which the Cloud Connector is supposed to connect.

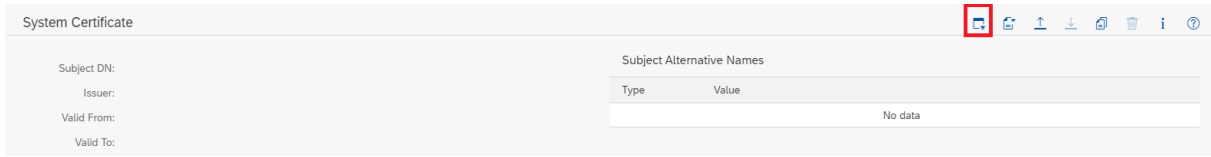
You must provide the system certificate as PKCS#12 file containing the client certificate, the corresponding private key and the CA root certificate that signed the client certificate (plus potentially the certificates of any intermediate CAs, if the certificate chain is longer than 2).

Procedure

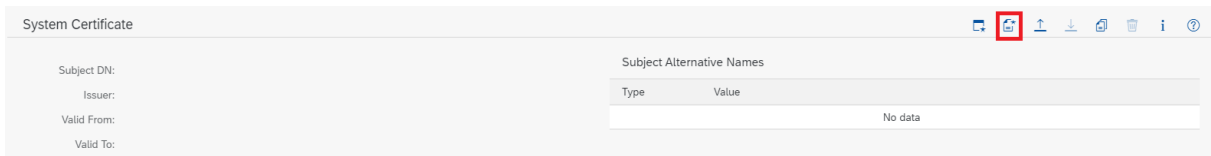
From the left panel, choose *Configuration*. On the tab *On Premise*, choose **System Certificate** **Import a certificate** to upload a certificate and provide its password:



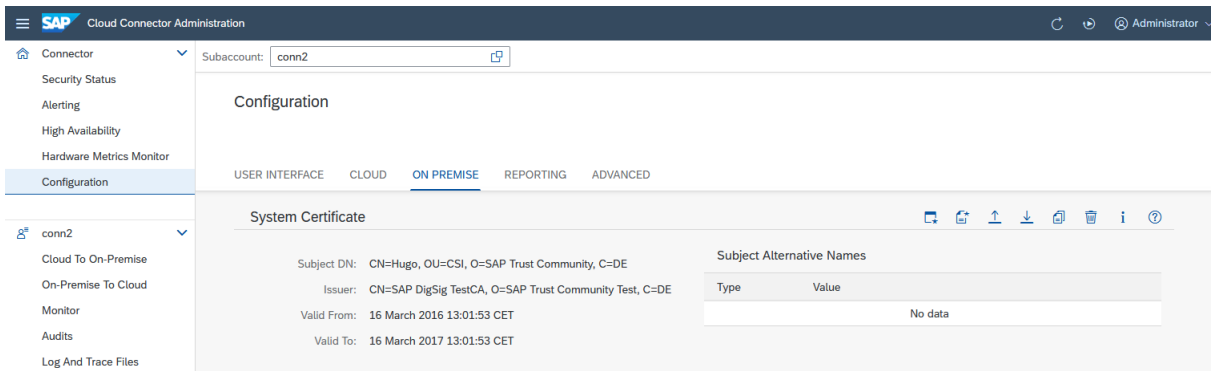
A second option is to start a *certificate signing request* procedure as described for the UI certificate in [Exchange UI Certificates in the Administration UI \[page 629\]](#) and upload the resulting signed certificate.



A third option is to generate a self-signed certificate. It might be useful if no CA is needed, for example, in a demo setup or if you want to use a dedicated CA. For this option, choose [Create and import a self-signed certificate](#):



If a system certificate has been imported successfully, its distinguished name, the name of the issuer, and the validity dates are displayed:



Delete, Export, or Renew a System Certificate

You can delete a system certificate that is no longer required. To do this, use the respective button and confirm deletion.

If you need the public key for establishing trust with a server, you can export the full chain via the [Export](#) button.

To renew a certificate that is close to expiration, install the new certificate as described above. This will replace the expiring certificate.

Related Information

[Configure Access Control \(HTTP\) \[page 457\]](#)

1.2.2.1.2 Initial Configuration (RFC)

Configure a Secure Network Connection (SNC) to set up the Cloud Connector for RFC communication to an ABAP backend system.

SNC Configuration for Mutual Authentication

To set up a mutual authentication between Cloud Connector and an ABAP backend system (connected via RFC), you can configure SNC for the Cloud Connector. It will then use the associated PSE for all RFC SNC requests. This means that the SNC identity, represented by this PSE, must:

- Be trusted by all backend systems to which the Cloud Connector is supposed to connect
- Play the role of a trusted external system by adding the SNC name of the Cloud Connector to the SNCSYSACL table. You can find more details in the SNC configuration documentation for the release of your ABAP system.

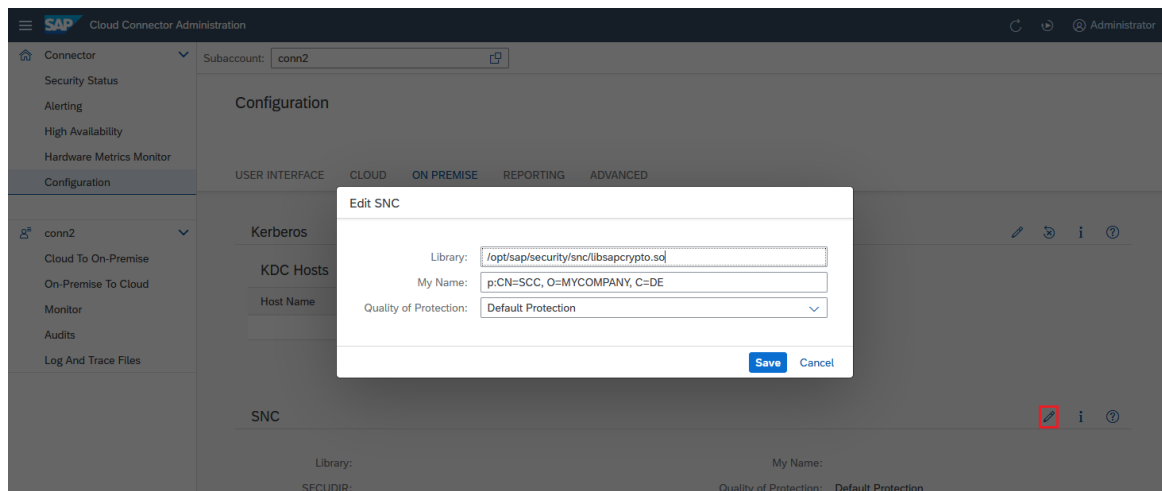
Prerequisites

- You have configured your ABAP system(s) for SNC. For detailed information on configuring SNC for an ABAP system, see also [Configuring SNC on AS ABAP](#).
- You have configured the ABAP System to trust the Cloud Connector's system SNC identity. To do this, or to establish trust for principal propagation, follow the steps described in [Configure Identity Propagation for RFC \[page 438\]](#).

Configuration Steps

1. Logon to the Cloud Connector
2. Choose *Configuration* from the main menu and go to tab *On Premise*, section *SNC*.
3. Enter the corresponding values in the fields *Library Name*, *My Name*, and *Quality of Protection*
4. Press *Save*.

Example:



- **Library Name:** Provides the location of the SNC library you are using for the Cloud Connector.

Note

Bear in mind that you must use one and the same security product on both sides of the communication.

- **My Name:** The SNC name that identifies the Cloud Connector. It represents a valid scheme for the SNC implementation that is used.
- **Quality of Protection:** Determines the level of protection that you require for the connectivity to the ABAP systems.

Note

When using **CommonCryptoLibrary** as SNC implementation, note [1525059](#) will help you to configure the PSE to be associated with the user running the Cloud Connector process.

Using the SAP Cryptographic Library

Note

This procedure is available as of Cloud Connector version 2.14.1

When using the SAP Cryptographic Library as SNC implementation, you can use the interactive setup scripts which can be found in the Cloud Connector's installation folder to ease the setup process.

Note

Using the scripts mentioned below is not mandatory for using the SAP cryptographic library. If you are familiar with the necessary steps needed for running a process with the desired identity, you can also configure the SNC setup for the SAP cryptographic library on your own.

Linux/Mac scripts: `snc_create_pse.sh`, `snc_import_ca_response.sh`

Windows scripts: `snc_create_pse.bat`, `snc_import_ca_response.bat`

1. Download and extract the SAP Cryptographic Library from the [Download Center](#) (search for `sapcryptolib`).
2. Copy the respective scripts depending on your OS to the SAP Cryptographic Library folder.
3. Make sure the Cloud Connector process is running.
4. Make sure the environment variable `SECUDIR` is properly set.
 1. For Linux, you can set it solely for the Cloud Connector process by extending the daemon as described in [Installation on Linux OS \[page 378\]](#).
5. Copy the partner's certificate you have [exported](#) to your `SECUDIR` folder. You must specify this to import it into your PSE.
6. Run the script `snc_create_pse` now. You should see the `SECUDIR` and the Cloud Connector user:
`SECUDIR D:\sec found.`
`Cloud Connector is running with user NT AUTHORITY\SYSTEM`
7. Specify the certificate name for the Cloud Connector:
`Specify the own certificate name in the format as CN=host,OU=org,O=comp,C=lang: CN=SCC`
`Specify the PSE password: <enter secure PW here>`
8. Specify the exported certificate of the partner from step 5:
`Specify the import certificate file of the partner contained in D:\sec: exported_abap_cert.cer`
`Creating PSE now...`
...
`PSE creation finished.`
9. Now, a PSE with name `scc.pse` has been created in your `SECUDIR` folder. Additionally, a file `sccCertificateRequest.p10` has been created. This is the CSR you can use to get a signed certificate by your CA now. You can import the response certificate from your CA directly, or use the other script `import_ca_response` later. If you don't want to sign your certificate by a CA and use it as a self-signed certificate, choose `no`. Depending on your CA, you might need to provide, besides the signed certificate, all other intermediate certificates and the root certificate for the import. Copy them to the `SECUDIR` folder and specify them in the order shown below.
`Do you want to import a CA response now? [y/n] y`
`Specify the CA response file contained in D:\sec: ca_response.crt`
`Specify further Root CAs (PEM, Base64 or DER binary) in D:\sec needed to complete the chain (separated by blanks), otherwise press enter: intermediate.crt root.crt`
`Importing CA response now...`
...
`CA-Response successfully imported into PSE "D:\sec\scc.pse"`
10. Now, all other required steps are done, such as creating the credentials file `cred_v2`, importing the partner certificate and exporting the Cloud Connector's certificate to `SECUDIR` with the name `scc.crt` (which must be [imported](#) at the partner's side).
`Create SSO server credentials...`
...
`Creation finished.`
`Export own certificate...`
...
`Export finished.`
`Import partner certificate into PSE file...`
...
`Completed.`

11. Restart the Cloud Connector and check the above screen if `SECUDIR` is set correctly. SNC setup for SAP Cryptographic Library should be complete now. In a next step, you must [Configure Access Control \[page 465\]](#) and, if needed, [Configure Identity Propagation for RFC \[page 438\]](#).

If you have further issues, check SAP note [1525059](#).

Related Information

[Configure Identity Propagation for RFC \[page 438\]](#)

1.2.2.1.3 Configuring the Cloud Connector for LDAP

Configure the Cloud Connector to support LDAP in different scenarios (cloud applications using LDAP or Cloud Connector authentication).

Prerequisites

You have installed the Cloud Connector and done the basic configuration:

[Installation \[page 349\]](#)

[Initial Configuration \[page 388\]](#)

Steps

When using LDAP-based user management, you have to configure the Cloud Connector to support this feature. Depending on the scenario, you need to perform the following steps:

Scenario 1: Cloud applications using LDAP for authentication. Configure the destination of the LDAP server in the Cloud Connector: [Configure Access Control \(LDAP\) \[page 472\]](#).

Scenario 2: Internal Cloud Connector user management. Activate LDAP user management in the Cloud Connector: [Use LDAP for User Administration \[page 634\]](#).

1.2.2.2 Managing Subaccounts

Add and connect your SAP BTP subaccounts to the Cloud Connector.

📘 Note

This topic refers to subaccount management in the Cloud Connector. If you are looking for information about managing subaccounts on SAP BTP (Cloud Foundry or Neo environment), see

- [Account Administration \(Cloud Foundry environment\)](#)
- [Administration and Operations, Neo Environment](#)

Context

You can connect to several subaccounts within a single Cloud Connector installation. Those subaccounts can use the Cloud Connector concurrently with different configurations. By selecting a subaccount from the drop-down box, all tab entries show the configuration, audit, and state, specific to this subaccount. In case of audit and traces, cross-subaccount info is merged with the subaccount-specific parts of the UI.

📘 Note

We recommend that you group only subaccounts with the same qualities in a single installation:

- Productive subaccounts should reside on a Cloud Connector that is used for productive subaccounts only.
- Test and development subaccounts can be merged, depending on the group of users that are supposed to deal with those subaccounts. However, the preferred logical setup is to have separate development and test installations.

Prerequisites

You have assigned one of these roles/role collections to the subaccount user that you use for initial Cloud Connector setup, depending on the SAP BTP environment in which your subaccount is running:

📘 Note

For the **Cloud Foundry** environment, you must know on which cloud management tools feature set (A or B) your account is running. For more information on feature sets, see [Cloud Management Tools — Feature Set Overview](#).

Environment	Required Roles/Role Collections	More Information
Cloud Foundry [feature set A]	<p>The user must be a <i>member of the global account</i> that the subaccount belongs to.</p> <p>Alternatively, you can assign the user as <i>Security Administrator</i>.</p>	<p>Add Members to Your Global Account</p> <p>Managing Security Administrators in Your Subaccount [Feature Set A]</p>
Cloud Foundry [feature set B]	<p>Assign at least one of these <i>default role collections</i> (all of them including the role <i>Cloud Connector Administrator</i>):</p> <ul style="list-style-type: none"> • Subaccount Administrator • Cloud Connector Administrator • Connectivity and Destination Administrator <p>Alternatively, you can assign a <i>custom role collection</i> to the user that includes the role <i>Cloud Connector Administrator</i>.</p>	<p>Default Role Collections [Feature Set B] [page 15]</p> <p>Role Collections and Roles in Global Accounts, Directories, and Subaccounts [Feature Set B]</p>
Neo	<p>Assign at least one of these <i>default roles</i>:</p> <ul style="list-style-type: none"> • Cloud Connector Admin • Administrator <p>Alternatively, you can assign a <i>custom role</i> to the user that includes the permission <code>manageSCCTunnels</code>.</p>	<p>Managing Member Authorizations in the Neo Environment</p>

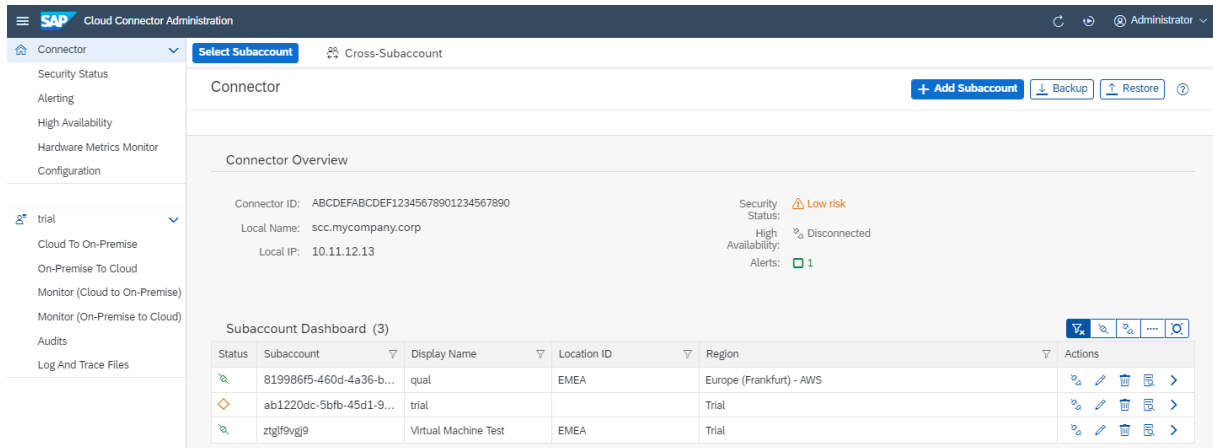
After establishing the Cloud Connector connection, this user is not needed any more, since it serves only for initial connection setup. You may revoke the corresponding role assignment then and remove the user from the [Members](#) list.

Note

If the Cloud Connector is installed in an environment that is operated by SAP, SAP provides a user that you can add as member in your SAP BTP subaccount and assign the required role.

Subaccount Dashboard

In the subaccount dashboard (choose your [Subaccount](#) from the main menu), you can check the state of all subaccount connections managed by this Cloud Connector at a glance.



In the screenshot above, the subaccount with display name *trial* (actual subaccount ID starts with *ab1220dc-5bfb-45d1-9...*) is already connected, but has no active resources exposed. All other subaccounts are connected with exposed resources and are fully operational. In addition, depending on the connection state, the dashboard allows you to do disconnect and connect subaccounts by pressing the respective button in the **Actions** column. You may also view details of, delete, or navigate to a subaccount using buttons from the **Actions** column.

The **Sort** buttons for the columns **Subaccount** and **Display Name** let you sort the entries by the column either ascending or descending, and the **Filter** buttons in the columns let you filter the listed entries.

You can use the **Filter** buttons above the dashboard to filter the shown subaccounts based on the connection status. You can select *all subaccounts*, *all connected ones*, *all disconnected ones*, *all subaccounts currently in connecting or reconnecting status*, or *all subaccounts for which establishing the connection has failed*.

If you want to connect an additional subaccount with your on-premise landscape, simply press the **Add Subaccount** button, which opens a dialog that is similar to the [Initial Configuration \[page 388\]](#) operation when establishing the first connection.

Add Subaccount

Region:*	<input type="text"/>
Subaccount:*	<input type="text"/>
Display Name:	<input type="text"/>
Subaccount User:*	<input type="text"/>
Password:*	<input type="password"/>
Location ID:	<input type="text" value="Enter location ID to overwrite default"/>
Description:	<input type="text"/>

Save

Cancel

Procedure

1. The <Region> field specifies the SAP BTP region that should be used, for example, **Europe (Rot)**. Choose the one you need from the drop-down list.

→ Remember

The available regions and region domains depend on the SAP BTP environment you are using. For more information, see [Regions](#) (Cloud Foundry and ABAP environment) or [Regions and Hosts Available for the Neo Environment](#).

→ Tip

You can also configure a region yourself, if it is not part of the standard list. Either insert the region host manually, or create a custom region, as described in [Configure Custom Regions \[page 418\]](#).

2. For <Subaccount> and <Subaccount User> (user/password), enter the values you obtained when you registered your account on SAP BTP.

ⓘ Note

If your subaccount is on **Cloud Foundry**, you must enter the subaccount ID as <Subaccount>, rather than its actual (technical) name. For information on getting the subaccount ID, see [Find Your](#)

[Subaccount ID \(Cloud Foundry Environment\) \[page 417\]](#). As `<Subaccount User>` you must provide your **Login E-mail** instead of a user ID.

For the **Neo** environment, enter the subaccount's **technical name** in the field `<Subaccount>`, not the subaccount ID.

Alternatively, you can add a new subaccount user in the SAP BTP cockpit, assign the required authorization (see section **Prerequisites** above), and use the new user and password.

Note

The Cloud Connector does not yet support *SAP Universal ID*. Please use your S-user or P-user credentials for the `<subaccount user>` and `<password>` fields instead.

For more information, see SAP note [3085908](#).

Tip

When using SAP Cloud Identity Services - Identity Authentication (IAS) as platform identity provider with two-factor authentication (2FA / MFA) for your subaccount, you can simply append the required token to the regular password. For example, if your password is "eX7?6rUm" and the one-time passcode is "123456", you must enter "eX7?6rUm123456" into the `<Password>` field.

Tip

For a subaccount in the **Cloud Foundry** environment, the Cloud Connector supports the use of a custom identity provider (IDP) via single sign-on (SSO) passcode. For more information, see [Use a Custom IDP for Subaccount Configuration \[page 409\]](#).

3. (Optional) You can define a `<Display Name>` that allows you to easily recognize a specific subaccount in the UI compared to the technical subaccount name.
4. (Optional) You can define a `<Location ID>` that identifies the location of this Cloud Connector for a specific subaccount. The location ID is used as routing information and therefore you can connect multiple Cloud Connectors to a single subaccount. If you don't specify any value for `<Location ID>`, the default is used, which represents the behavior of previous Cloud Connector versions. 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.
5. (Optional) You can provide a `<Description>` of the subaccount that is shown when clicking on the *Details* icon in the *Actions* column.
6. Choose *Save*.

Next Steps

- To modify an existing subaccount, choose the *Edit* icon and change the `<Display Name>`, `<Location ID>` and/or `<Description>`.

Edit Subaccount

Display Name:	<input type="text" value="test"/>
Location ID:	<input type="text" value="Enter location ID to overwrite default"/>
Description:	<input type="text"/>

[Save](#) [Cancel](#)

- You can also delete a subaccount from the list of connections. The subaccount will be disconnected and all configurations will be removed from the installation.

Related Information

[Managing Member Authorizations in the Neo Environment](#)

[Copy a Subaccount Configuration \[page 407\]](#)

[Update the Certificate for a Subaccount \[page 413\]](#)

[Configure a Disaster Recovery Subaccount \[page 415\]](#)

[Find Your Subaccount ID \(Cloud Foundry Environment\) \[page 417\]](#)

[Configure Custom Regions \[page 418\]](#)

[Use a Custom IDP for Subaccount Configuration \[page 409\]](#)

1.2.2.2.1 Copy a Subaccount Configuration

Copy an existing subaccount configuration in the Cloud Connector to another subaccount.

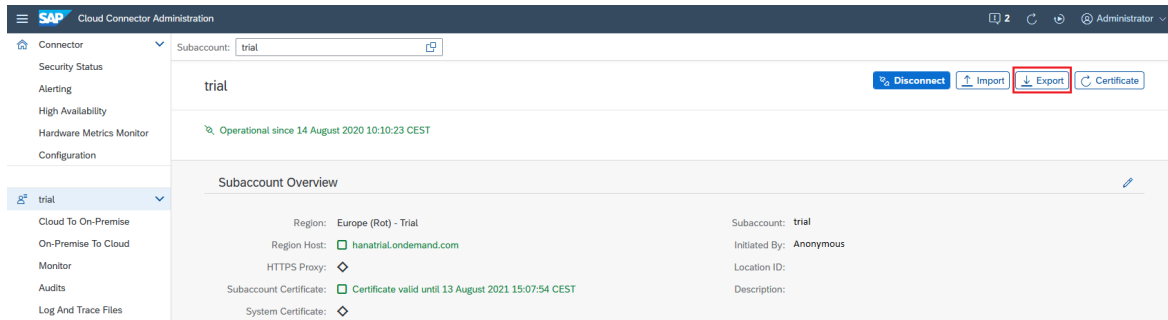
You can copy the configuration of a subaccount's *Cloud To On-Premise* and *On-Premise To Cloud* sections to a new subaccount, by using the export and import functions in the Cloud Connector administration UI.

Note

Principal propagation configuration (section *Cloud To On-Premise*) is not exported or imported, since it contains subaccount-specific data.

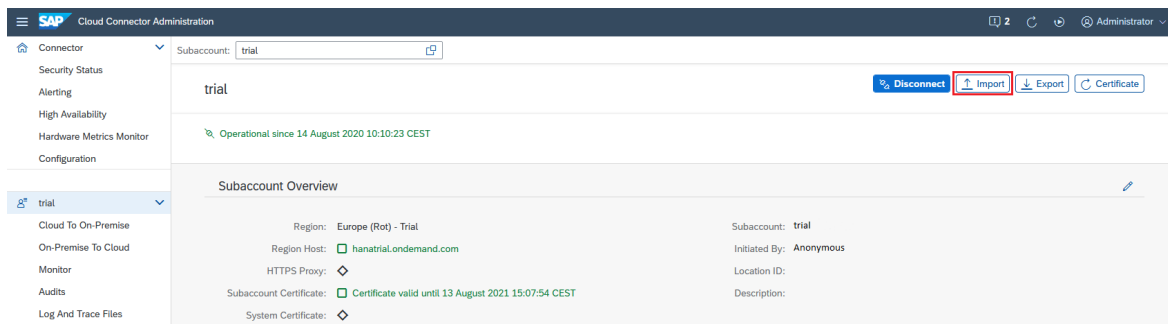
Procedure: Export an Existing Configuration

1. In the Cloud Connector administration UI, choose your subaccount from the navigation menu.
2. To export the existing configuration, choose the *Export* button in the upper right corner. The configuration is downloaded as a zip file to your local file system.



Procedure: Import an Existing Configuration

1. From the navigation menu, choose the subaccount to which you want to copy an existing configuration.
2. To import an existing configuration, choose the *Import* button in the upper right corner.



3. Select one of the following sources:
 1. *File*, if you want to copy the configuration from a previously downloaded zip file.
 2. *Subaccount*, if you want to copy the configuration directly from another existing subaccount.

Import Account Configuration

Source: File

Subaccount

File:

Browse

Import

Cancel

1.2.2.2.2 Use a Custom IDP for Subaccount Configuration

Enable custom identity provider (IDP) authentication to configure a Cloud Foundry subaccount in the Cloud Connector by using a one-time passcode.

Content

[Context \[page 409\]](#)

[Get the URL \[Feature Set A\] \[page 412\]](#)

[Get the URL \[Feature Set B\] \[page 413\]](#)

[Get the One-Time Passcode \[page 413\]](#)

Context

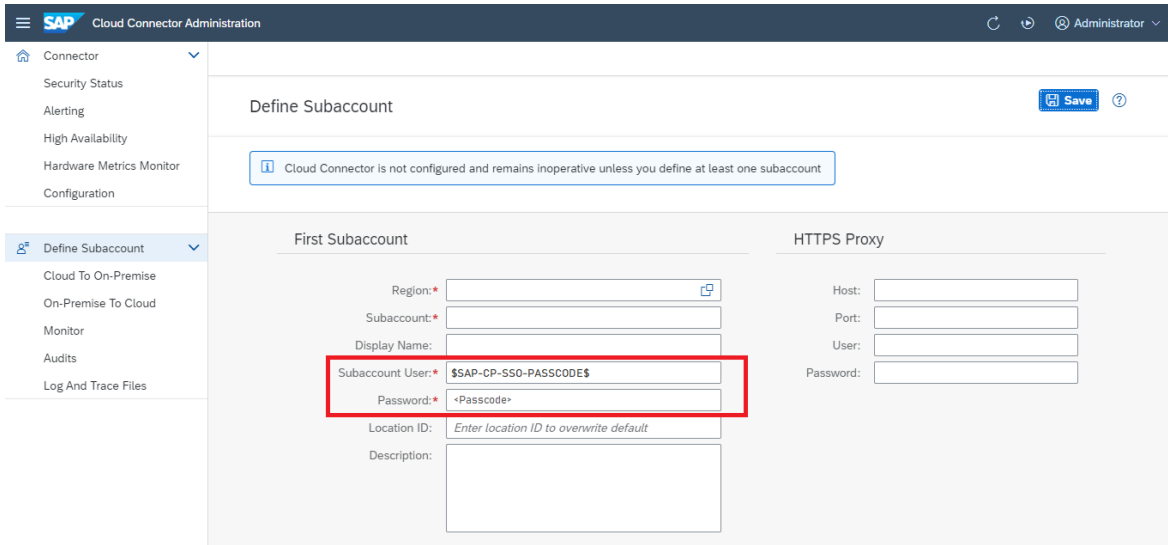
For a subaccount in the **Cloud Foundry** environment that uses a custom IDP, you can choose this IDP for authentication instead of the (default) SAP ID service when configuring the subaccount in the Cloud Connector.

Using custom IDP authentication, you can perform the following operations in the Cloud Connector:

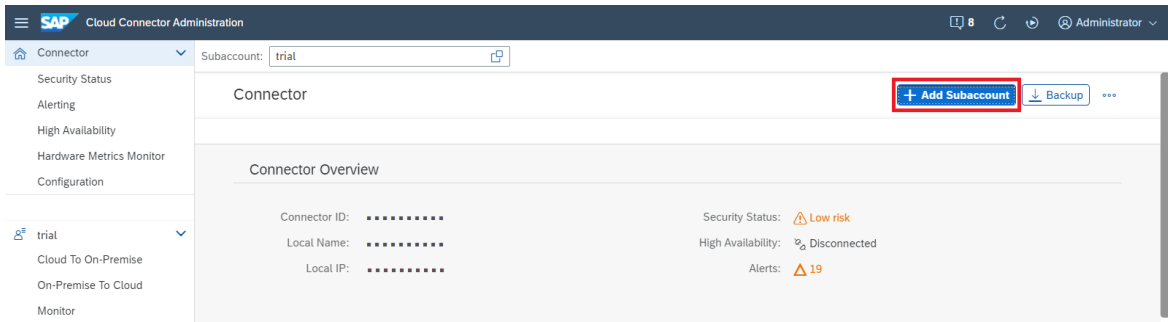
Operation	Description
Set up Connection Parameters and HTTPS Proxy [page 391]	Add an <i>initial</i> subaccount to a fresh Cloud Connector installation.
Managing Subaccounts [page 401]	Add <i>additional</i> subaccounts to an existing Cloud Connector installation.
Update the Certificate for a Subaccount [page 413]	Refresh a subaccount certificate's validity period.

To enable custom IDP authentication, for each of these operations you must enter the marker value **\$SAP-CP-SSO-PASSCODE\$** in the `<Subaccount User>` or `<User Name>` field, and a one-time generated passcode (known as *temporary authentication code*) in the `<Password>` field:

- When adding the **initial subaccount** to a fresh Cloud Connector installation, enter the user name **\$SAP-CP-SSO-PASSCODE\$** and the passcode on the Cloud Connector's *Define Subaccount* screen (see also [Set up Connection Parameters and HTTPS Proxy \[page 391\]](#)):



- When adding one or more **additional subaccount(s)** to an existing Cloud Connector installation, provide the user name **\$\$AP-CP-SSO-PASSCODE\$** and the passcode via the *Connector* screen (see also [Managing Subaccounts \[page 401\]](#)):



Add Subaccount

Region:*

Subaccount:*

Display Name:

Subaccount User:*

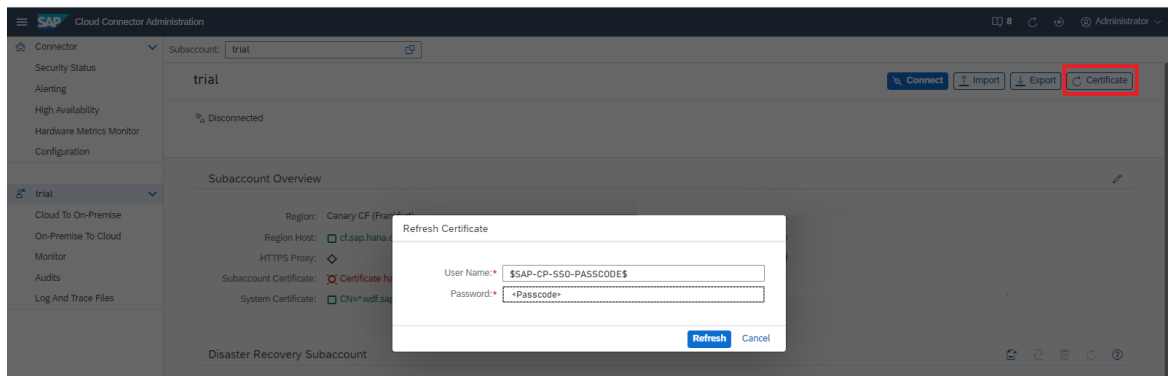
Password:*

Location ID:

Description:

Save Cancel

- To **refresh a subaccount certificate**, enter the user name **\$SAP-CP-SSO-PASSCODE\$** and the passcode via the corresponding **<Subaccount>** screen (see also [Update the Certificate for a Subaccount \[page 413\]](#)):



To retrieve the one-time generated passcode, you must use the correct login URL for single sign-on (SSO) to access your custom IDP. The procedure to get this URL depends on the SAP BTP feature set you are using.

Note

To choose the right procedure, you must know on which cloud management tools feature set (A or B) your SAP BTP account is running. For more information on feature sets, see [Cloud Management Tools – Feature Set Overview](#).

Next Step

[Get the URL \[Feature Set A\] \[page 412\]](#)

[Get the URL \[Feature Set B\] \[page 413\]](#)

⚠ Caution

Mind the respective user rights described in the prerequisites for [Initial Configuration \[page 388\]](#) and [Managing Subaccounts \[page 401\]](#). For feature set A, it is mandatory to be a subaccount *Security Administrator*, not only a global account member.

Back to [Content \[page 409\]](#)

Get the URL [Feature Set A]

Choose one of the following options:

Option 1: Assemble the URL

The URL pattern is `https://login.cf.<btp-region-host>/passcode`.

1. Get the SAP BTP region host, for example, `eu10.hana.ondemand.com`.
2. Assemble the final URL to be used, in this case: `https://login.cf.eu10.hana.ondemand.com/passcode`.

Option 2: Get the URL Using the Cloud Foundry CLI

Use the Cloud Foundry [CLI](#)  to perform the following steps:

1. Execute the command `cf api` to navigate to the SAP BTP region.
2. Execute `cf login --sso` to get the URL.

🔗 Sample Code

```
$ cf api api.cf.eu10.hana.ondemand.com
Setting api endpoint to api.cf.eu10.hana.ondemand.com...
OK
api endpoint: https://api.cf.eu10.hana.ondemand.com
api version: 2.156.0
$ cf login --sso
API endpoint: https://api.cf.eu10.hana.ondemand.com
Temporary Authentication Code ( Get one at https://
login.cf.eu10.hana.ondemand.com/passcode ):
```

Next Step

[Get the One-Time Passcode \[page 413\]](#)

Back to [Content \[page 409\]](#)

Get the URL [Feature Set B]


Choose one of the following options:

Option 1: Assemble the URL

The URL pattern is `https://<subdomain>.authentication.<btp-XSUA-host>/passcode`.

1. Get the SAP BTP region host, for example, `eu10.hana.ondemand.com`.
2. Assemble the final URL to be used, in this case: `https://mysubdomain.authentication.eu10.hana.ondemand.com/passcode`.

Option 2: Get the URL Using the Connectivity Service Instance Credentials

1. Choose one of these steps to obtain the URL:
 - [Create and Bind a Connectivity Service Instance \[page 206\]](#)
 - [Create a service key](#) 
2. Get the value of the `token_service_url` attribute.
3. Append `/passcode` at the end of the obtained URL.

Next Step

[Get the One-Time Passcode \[page 413\]](#)

Back to [Content \[page 409\]](#)

Get the One-Time Passcode

1. Open the resulting URL in your browser to get the one-time passcode via SSO:
 - If *there is* an active user session, the passcode is generated automatically and returned right away.
 - If there is *no* active user session, you are asked to log on to the IDP manually. If several IDPs are configured, you can choose one from the available options.
2. Use the passcode to proceed with the subaccount configuration in the Cloud Connector UI.
Back to [Content \[page 409\]](#)

1.2.2.2.3 Update the Certificate for a Subaccount

Certificates used by the Cloud Connector are issued with a limited validity period. To prevent a downtime while refreshing the certificate, you can update it for your subaccount directly from the administration UI.

Prerequisites

You must have the required subaccount authorizations on SAP BTP to update certificates for your subaccount.

See:

- [Connectivity: Technical Roles \[page 11\]](#) (Cloud Foundry environment)

Procedure

→ Tip

You can use this procedure even if the certificate has already expired.

Proceed as follows to update your subaccount certificate:

1. From the main menu, choose your subaccount.

Note

To check the certificate's validity, click on the [<Subaccount Certificate>](#) in section [Subaccount Overview](#).

2. Choose the [Certificate](#) button. A dialog opens, requesting a user name and password.
3. Enter [<User Name>](#) and [<Password>](#) and choose [OK](#). The certificate assigned to your subaccount is refreshed.

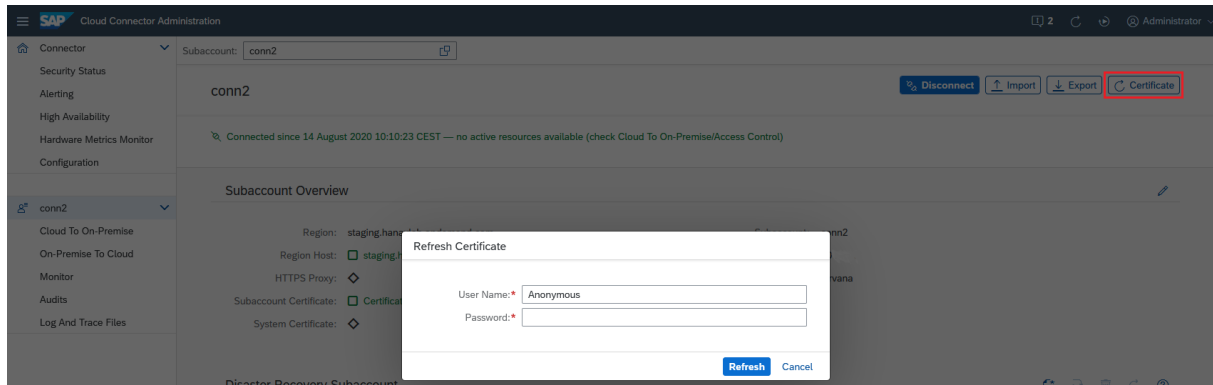
Note

In the **Cloud Foundry** environment, you must provide your [Login E-mail](#) instead of a user ID as [<User Name>](#).

→ Tip

When using SAP Cloud Identity Services - Identity Authentication (IAS) as platform identity provider with two-factor authentication (2FA / MFA) for your subaccount, you can simply append the required token to the regular password. For example, if your password is "eX7?6rUm" and the one-time passcode is "123456", you must enter "eX7?6rUm123456" into the [<Password>](#) field.

4. If you have configured a disaster recovery subaccount, go to section [Disaster Recovery Subaccount](#) below and choose [Refresh Disaster Recovery Certificate](#).
5. Enter [<User Name>](#) and [<Password>](#) as in step 3 and choose [OK](#).



1.2.2.2.4 Configure a Disaster Recovery Subaccount

Configure a subaccount as backup for disaster recovery.

⚠ Caution

This feature is deprecated.

Due to the discontinuation of the *Enhanced Disaster Recovery Service*, the related functionality in the Cloud Connector has been dropped as of version 2.16.

For more information, see [What's New for SAP Business Technology Platform](#).

Each subaccount (except trial accounts) can optionally have a disaster recovery subaccount.

Prerequisite is that you are using the enhanced disaster recovery.

The disaster recovery subaccount is intended to take over if the region host of its associated original subaccount faces severe issues.

A disaster recovery account inherits the configuration from its original subaccount except for the region host. The user can, but does not have to be the same.

Procedure

1. From the main menu, choose your subaccount.
2. In section *Disaster Recovery Subaccount*, choose *Configure disaster recovery subaccount*.
3. In the configuration dialog, select an appropriate *<Region Host>* from the drop-down list.

📘 Note

The selected region host must be different from the region host of the original subaccount.

4. (Optional) You can adjust the `<Subaccount User>`.
5. Enter the `<Password>` for the subaccount user.
6. If configured, enter a `<Location ID>`.
7. Choose *Save*.

Note

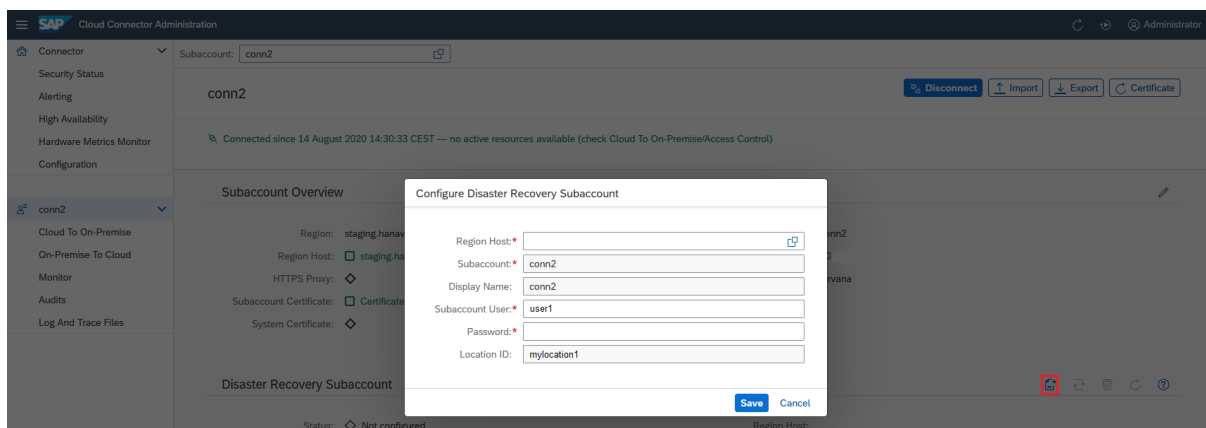
The technical subaccount name, the display name, and the location ID must remain the same. They are set automatically and cannot be changed.

Note

You cannot choose another original subaccount nor a trial subaccount to become a disaster recovery subaccount.

Note

If you want to change a disaster recovery subaccount, you must delete it first and then configure it again.



To switch from the original subaccount to the disaster recovery subaccount, choose *Employ disaster recovery subaccount*.

The disaster recovery subaccount then becomes active, and the original subaccount is deactivated.

You can switch back to the original subaccount as soon as it is available again.

Note

As of Cloud Connector 2.11, the cloud side informs about a disaster by issuing an event. In this case, the switch is performed automatically.

Related Information

[Convert a Disaster Recovery Subaccount into a Standard Subaccount \[page 417\]](#)

1.2.2.2.4.1 Convert a Disaster Recovery Subaccount into a Standard Subaccount

Convert a disaster recovery subaccount into a standard subaccount if the former primary subaccount's region cannot be recovered.

⚠ Caution

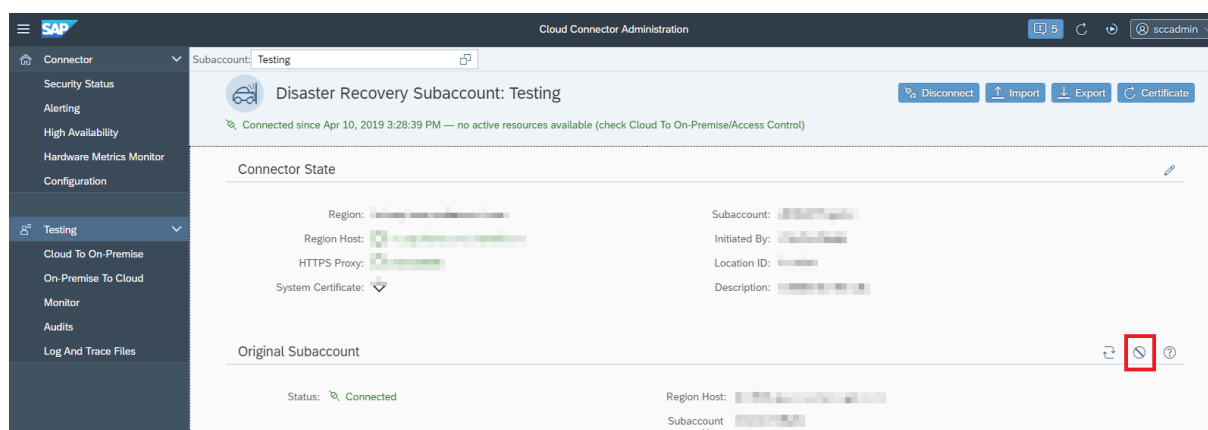
This feature is deprecated.

Due to the discontinuation of the *Enhanced Disaster Recovery Service*, the related functionality in the Cloud Connector has been dropped as of version 2.16.

For more information, see [What's New for SAP Business Technology Platform](#).

Disaster recovery subaccounts that were switched to disaster recovery mode can be elevated to standard subaccounts if a disaster recovery region replaces an original region that is not expected to recover.

If a disaster recovery subaccount should be used as primary subaccount, you can convert it by choosing the button *Discard original subaccount and replace it with disaster recovery subaccount*.



1.2.2.2.5 Find Your Subaccount ID (Cloud Foundry Environment)

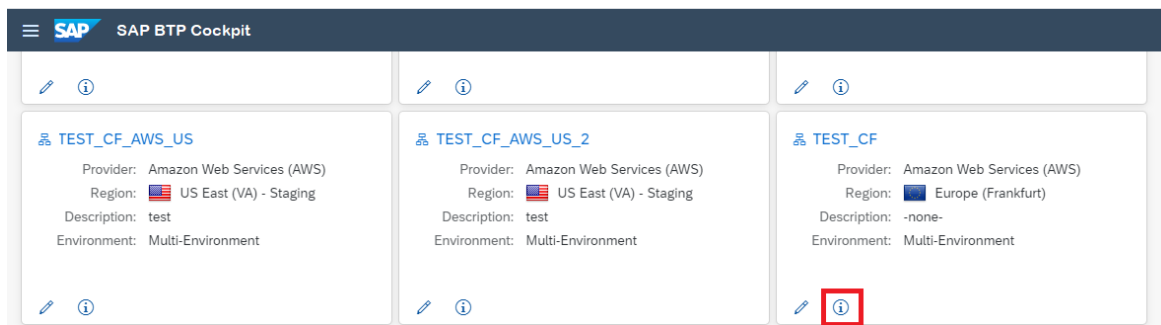
Get your subaccount ID to configure the Cloud Connector in the Cloud Foundry environment.

📌 Note

For the Beta version, the cloud cockpit is not yet available.

In order to set up your subaccount in the Cloud Connector, you must know the subaccount ID. Follow these steps to acquire it:

1. Open the SAP BTP cockpit.
2. Navigate to the subaccount list of the global account containing your subaccount: choose **Home** **>** **<Your Global Account>** **>** **Account Explorer**.
3. Find your subaccount in the list.
4. Choose the **Info** icon in the subaccount tile to display the subaccount ID:



1.2.2.2.6 Configure Custom Regions

Configure regions that are not available in the standard selection.

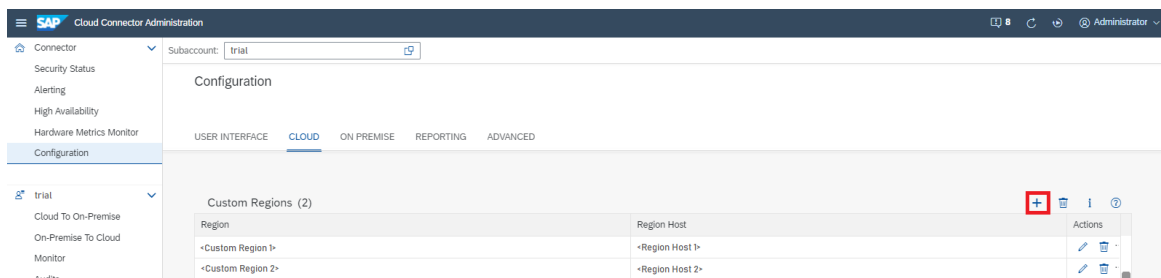
→ Tip

This procedure is useful in particular for regions introduced after the release of your current Cloud Connector version. Those regions are not included in the list of predefined regions.

If you want to use a custom region for your subaccount, you can configure regions in the Cloud Connector, which are not listed in the selection of standard regions.

To add a custom region, do the following:

1. From the Cloud Connector main menu, choose **Configuration** **>** **Cloud** and go to the **Custom Regions** section.
2. To add a region to the list, choose the **Add** icon.



3. In the **Add Region** dialog, enter the **<Region>** and **<Region Host>** you want to use.
4. Choose **Save**.

5. To edit a region from the list, select the corresponding line and choose the *Edit* icon.

1.2.2.3 Authenticating Users against On-Premise Systems

Authentication types supported by the Cloud Connector.

Currently, the Cloud Connector supports **basic authentication**, **principal propagation**, and **technical user propagation** as user authentication types towards internal systems. The destination configuration of the used cloud application defines which of these types is used for the actual communication to an on-premise system through the Cloud Connector, see [Managing Destinations \[page 59\]](#).

- To use **basic authentication**, configure an on-premise system to accept basic authentication and to provide one or multiple service users. No additional steps are necessary in the Cloud Connector for this authentication type.
- To use **principal propagation** or **technical user propagation**, you must explicitly configure trust to those cloud entities from which user tokens are accepted as valid.
For more information, see [Configuring Principal Propagation \[page 420\]](#) and [Configuring Technical User Propagation \[page 455\]](#).
- When using HTTP as communication protocol, you can also perform a logon using the Cloud Connector system certificate, if there is no principal provided by the cloud side. The access control entry must be configured for this.

Note

For HTTP, also some other token-based authentication methods might work between the cloud application and the backend, depending on the backend capabilities.

Related Information

[Configuring Principal Propagation \[page 420\]](#)

[Configuring Technical User Propagation \[page 455\]](#)

1.2.2.3.1 Configuring Principal Propagation

Use principal propagation to simplify the access of SAP BTP users to on-premise systems.

Tasks in this section:

Task	Description
Set Up Trust [page 420]	Configure a trusted relationship in the Cloud Connector to support principal propagation. Principal propagation lets you forward the logged-on identity in the cloud to the internal system without requesting a password.
Configure a CA Certificate [page 424]	Install and configure an X.509 certificate to enable support for principal propagation.
Configuring Identity Propagation to an ABAP System [page 429]	Learn more about the different types of configuring and supporting principal propagation for a particular AS ABAP.
Configure Subject Patterns for Principal Propagation [page 443]	Define a pattern identifying the user for the subject of the generated short-lived X.509 certificate, as well as its validity period.
Configure a Secure Login Server [page 447]	Configuration steps for Java Secure Login Server (SLS) support.
Configure Kerberos [page 451]	The Cloud Connector lets you propagate users authenticated in SAP BTP via Kerberos against back-end systems. It uses the Service For User and Constrained Delegation protocol extension of Kerberos.
Configuring Identity Propagation to SAP NetWeaver AS for Java [page 453]	Find step-by-step instructions on how to configure principal propagation to an application server Java (AS Java).

Related Information

[Principal Propagation \[page 168\]](#) (Cloud Foundry environment)

[Principal Propagation](#) (Neo environment)

1.2.2.3.1.1 Set Up Trust

Establish trust to an identity provider to support principal propagation and technical user propagation.

Tasks

[Configure Trusted Entities in the Cloud Connector \[page 421\]](#)

[Configure an On-Premise System for Principal Propagation \[page 422\]](#)

[Trust Cloud Applications in the Cloud Connector \[page 423\]](#)

[Set up a Trust Store \[page 423\]](#)

Configure Trusted Entities in the Cloud Connector

Note

The information in this section applies to both principal propagation and technical user propagation.

You perform trust configuration to support principal propagation, that is, forwarding the *logged-on identity* in the cloud to the internal system without the need of providing the password. The same is done for technical user propagation, which logs on a *technical* user identified by an access token for an OAuth client without the need of providing the password.

By default, your Cloud Connector does not trust any entity that issues tokens for principal propagation. Therefore, the list of trusted identity providers (IdPs) is empty by default.

If you decide to use the principal propagation feature, you must establish trust to at least one IdP. The following IdP types are supported:

- **Neo** environment: *SAML* IdPs.

Note

In the Neo environment, you can also trust *HANA instances* and *Java applications* to act as IdPs.

- **Cloud Foundry** environment: *OAuth* IdPs.

You can configure trust to one or more IdPs per subaccount. After you've configured trust in the cockpit for your subaccount, for example, to your own company's IdP(s), you can synchronize this list with your Cloud Connector.

The screenshot shows the SAP Cloud Connector Administration interface. The top navigation bar includes the SAP logo and the text 'Cloud Connector Administration'. Below the navigation bar, there is a sidebar on the left with a menu containing 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. The main content area is titled 'Cloud To On-Premise' and has a subaccount dropdown set to 'conn3'. The 'PRINCIPAL PROPAGATION' tab is selected, showing a 'Trust Configuration (0)' section. This section contains a table with columns for 'Name', 'Description', 'Type', 'Trusted', and 'Actions'. The table is currently empty, with a message below it: 'Synchronize to obtain a list of IdPs or applications'.

From your subaccount menu, choose [Cloud to On-Premise](#) and go to the [Principal Propagation](#) tab. Choose the [Synchronize](#) button to store the list of existing identity providers locally in your Cloud Connector.

Select an entry to see its details:

- **Name:** the name associated with the identity provider.
- **Description:** descriptive information about this entry.
- **Type:** type of the trusted entity.
- **Trusted:** indicates whether the entry is trusted for principal propagation.
- **Actions:** Choose the [Show Certificate Information](#) icon to display detail information for the corresponding entry. The Cloud Connector runtime will use the certificate associated with the entry to verify that the assertion used for principal propagation was issued by a trusted entity.

You can decide for each entry, whether to trust it for the principal propagation use case by choosing [Edit](#) and (de)selecting the [Trusted](#) checkbox.

Note

Whenever you update a SAML IdP configuration for a subaccount on cloud side, you must synchronize the trusted entities in the Cloud Connector. Otherwise the validation of the forwarded SAML assertion will fail with an exception containing an exception message similar to this:

Caused by: com.sap.engine.lib.xml.signature.SignatureException: Unable to validate signature -> java.security.SignatureException: Signature decryption error: javax.crypto.BadPaddingException: Invalid PKCS#1 padding: encrypted message and modulus lengths do not match!.

For more information, see also [Include Tokens from Corporate Identity Providers or Identity Authentication in Tokens of the SAP Authorization and Trust Management Service](#).

Back to [Tasks \[page 421\]](#)

Configure an On-Premise System for Principal Propagation

Set up principal propagation from SAP BTP to your internal system that is used in a hybrid scenario.

Note

As a prerequisite for principal propagation for RFC, the following cloud application runtime versions are required:

- for Java Web: **1.51.8** or higher
- for Java EE 6 Web Profile: **2.31.11** or higher
- other runtimes support it with any version

1. Set up trust to an entity that is issuing an assertion for the logged-on user (see section above).
2. Set up the system identity for the Cloud Connector.
 - For HTTPS, you must import a system certificate into your Cloud Connector.
 - For RFC, you must import an SNC PSE into your Cloud Connector.

3. Configure the target system to trust the Cloud Connector.

There are two levels of trust:

1. First, you must allow the Cloud Connector to identify itself with its system certificate (for HTTPS), or with the SNC PSE (for RFC).
2. Then, you must allow this identity to propagate the user accordingly:
 - For HTTPS, the Cloud Connector forwards the true identity in a short-lived x.509 certificate in an HTTP header named **SSL_CLIENT_CERT**. The system must use this certificate for logging on the real user. The SSL handshake, however, is performed through the system certificate. For more information on identity forwarding, see [Configure Access Control \(HTTP\) \[page 457\]](#).
 - For RFC, the Cloud Connector forwards the true identity as part of the RFC protocol.

For more information, see [Configuring Identity Propagation to an ABAP System \[page 429\]](#).

4. Configure the user mapping in the target system. The x.509 certificate contains information about the cloud user in its subject. Use this information to map the identity to the appropriate user in this system. This step applies for both HTTPS and RFC.

📌 Note

If you have the following scenario: *Application1->AppToAppSSO->Application2->Principal Propagation->On premise Backend System* you must mark *Application2* as **trusted** by the Cloud Connector in tab *Principal Propagation*, section *Trust Configuration*.

If you use an identity provider that issues unsigned assertions, you must mark all relevant applications as **trusted** by the Cloud Connector in tab *Principal Propagation*, section *Trust Configuration*.

Back to [Tasks \[page 421\]](#)

Trust Cloud Applications in the Cloud Connector

Configure an allowlist for trusted cloud applications, see [Configure Trust \[page 616\]](#).

Back to [Tasks \[page 421\]](#)

Set up a Trust Store

Configure a trust store that acts as an allowlist for trusted on-premise systems. See [Configure Trust \[page 616\]](#).

Back to [Tasks \[page 421\]](#)

Related Information

[Principal Propagation \[page 168\]](#) (Cloud Foundry)

[Principal Propagation](#) (Neo)

1.2.2.3.1.2 Configure a CA Certificate

Install and configure an X.509 certificate to enable support for principal propagation in the Cloud Connector.

Note

The information in this section applies to both principal propagation and technical user propagation.

Supported CA Mechanisms

You can enable support for principal propagation or technical user propagation with X.509 certificates by performing either of the following procedures:

- Using a *Local CA* in the Cloud Connector.

Note

Prior to version 2.7.0, this was the only option and the system certificate was acting both as client certificate and CA certificate in the context of principal propagation.

- Using a *Secure Login Server (SLS)* and delegate the CA functionality to it.

The Cloud Connector uses the configured CA approach to issue short-lived certificates for logging on the same identity in the back end that is logged on in the cloud. For establishing trust with the back end, the respective configuration steps are independent of the approach that you choose for the CA.

Install a local CA Certificate

To issue short-lived certificates that are used for principal propagation to a back-end system, you can import an X.509 client certificate into the Cloud Connector. This CA certificate must be provided as *PKCS#12* file containing the (intermediate) certificate, the corresponding private key, and the CA root certificate that signed the intermediate certificate (plus the certificates of any other intermediate CAs, if the certificate chain includes more than those two certificates).

Use either of the following options to install a local CA certificate:

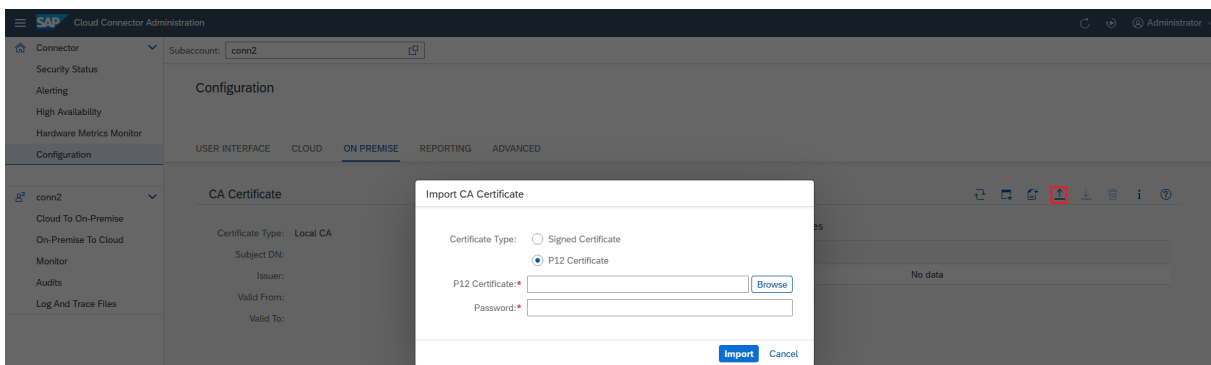
- Option 1: Choose the PKCS#12 file from the file system, using the file upload dialog. For the import process, you must also provide the file password.

- Option 2: Start a *Certificate Signing Request (CSR)* procedure like for the UI certificate, see [Exchange UI Certificates in the Administration UI \[page 629\]](#).
- Option 3: Generate a self-signed certificate, which might be useful in a demo setup or if you need a dedicated CA. In particular for this option, it is useful to export the public key of the CA via the button *Download certificate in DER format*.

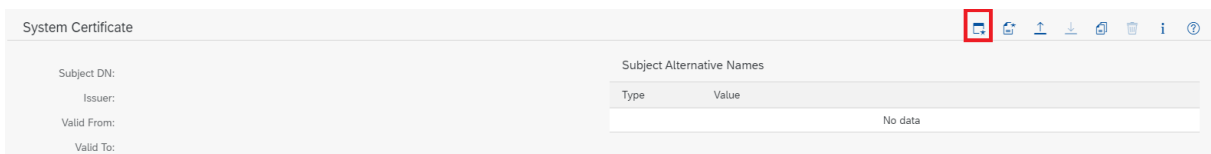
⚠ Caution

The CA certificate should have the `keyUsage` attribute `keyCertSign`. Many systems verify that the issuer of a certificate has this attribute and deny a client certificate, if this attribute is not present. When using the *Certificate Signing Request* procedure, the attribute will be requested for the CA certificate. Also, when generating a self-signed certificate, this attribute will be added automatically.

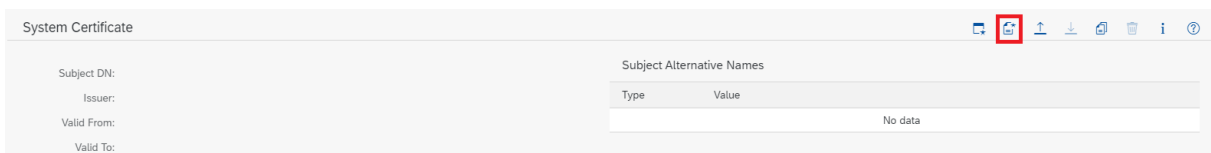
Choose *Import a certificate* for option 1:



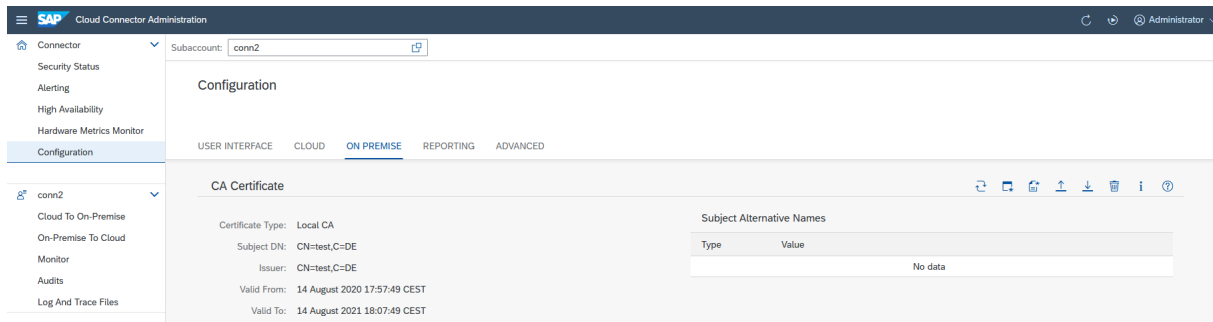
Choose *Generate a certificate signing request* for option 2:



Choose *Create and import a self-signed certificate* if you want to use option 3:



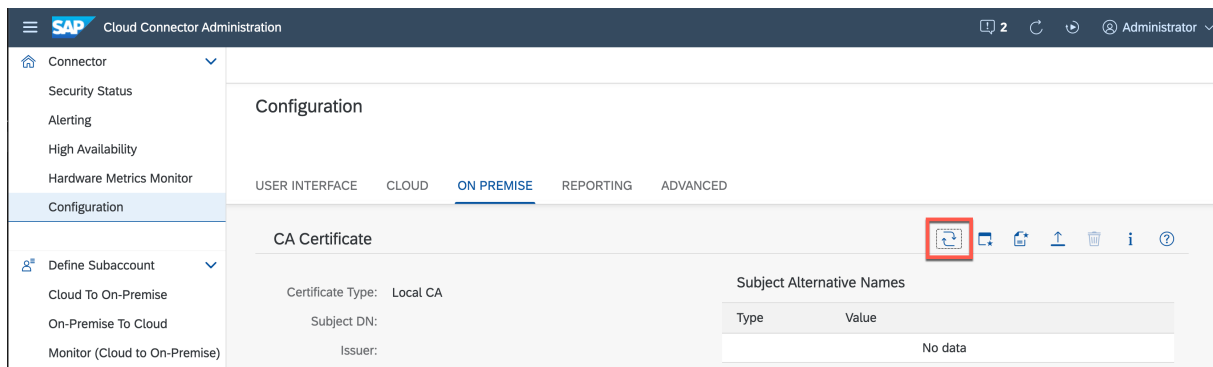
In particular for this option, it is useful to export the public key of the CA by choosing the respective button. After successful import of the CA certificate, its distinguished name, the name of the issuer, and the validity dates are shown:



If a CA certificate is no longer required, you can delete it. Use the respective [Delete](#) button and confirm the deletion.

Configure a CA Hosted by a Secure Login Server

If you want to delegate the CA functionality to a *Secure Login Server* (SLS), choose the CA using the [Secure Login Server](#) option and configure the SLS as follows, after having configured the SLS as described in [Configure a Secure Login Server \[page 447\]](#).



A wizard offers in a first step a quick configuration by metadata URL pointing to the SLS you'd like to use.

Configure CA Via Secure Login Server

i You may leave this field empty and go to the next step to set up the configuration without metadata URL

Meta URL:

[Previous](#)
Next
[Cancel](#)

Using the metadata URL lets you fetch the most relevant data from SLS instance. You only have to choose the profile configured on SLS, which should be used for the generation of short-lived certificates. Choose *Finish* to save the configuration.

Configure CA Via Secure Login Server

Profile ID: *

.....

.....

.....

[Previous](#)
Finish
[Cancel](#)

Note
The URL won't be stored in the Cloud Connector configuration.

If you don't have the metadata URL or would change the current configuration without metadata URL, you may keep the field empty and go to the next step.

In that case, you must provide all configuration details manually.

Enter the following:

- **API version:** Version of the SLS configuration protocol.

- `Host Name`: Host on which your SLS is installed.
- `Port`: Port for communication with SLS during the configuration.
- `Profile ID`: SLS profile ID that allows to issue certificates as needed for principal propagation with the Cloud Connector.

Note

Used if SLS API *version 3* is configured.

- `Profile`: SLS profile that allows to issue certificates as needed for principal propagation with the Cloud Connector.

Note

Used if SLS API *version 2* is configured.

- `Authentication Port`: Port over which the Cloud Connector is requesting the short-lived certificates from SLS.

Note

For this privileged port, a client certificate authentication is required, for which the Cloud Connector's system certificate is used.

Configure CA Via Secure Login Server

API version: *

Host Name: *

Port:

Previous
Next
Cancel

In the next step, you can finalize the configuration. The fields in the next step depend on the chosen SLS API version.

Configure CA Via Secure Login Server

Profile ID: *

Authentication Port:

[Previous](#)
Finish
[Cancel](#)

Choose *Finish* to save the configuration.

Related Information

[Configure a Secure Login Server \[page 447\]](#)

[Initial Configuration \(HTTP\) \[page 396\]](#)

[Initial Configuration \(RFC\) \[page 398\]](#)

1.2.2.3.1.3 Configuring Identity Propagation to an ABAP System

Learn more about the different types of configuring and supporting principal propagation and technical user propagation for a particular AS ABAP.

Note

The information in this section applies to both principal propagation and technical user propagation.

Task	Description
Configure Identity Propagation for HTTPS [page 430]	Step-by-step instructions to configure principal propagation to an ABAP server for HTTPS.
Configure Identity Propagation via SAP Web Dispatcher [page 435]	Set up a trust chain to use principal propagation to an ABAP server for HTTPS via SAP Web Dispatcher.

Task	Description
Configure Identity Propagation for RFC [page 438]	Step-by-step instructions to configure principal propagation to an ABAP server for RFC.
Rule-Based Mapping of Certificates [page 441]	Map short-lived certificates to users in the ABAP server.

1.2.2.3.1.3.1 Configure Identity Propagation for HTTPS

Find step-by-step instructions to configure principal propagation to an ABAP server for HTTPS.

Note

The information in this section applies to both principal propagation and technical user propagation. These two types of user propagation are combined as *identity* propagation.

Example Data

The following data are used in this example:

- System certificate was issued by: **CN=MyCompany CA,O=Trust Community,C=DE**.
- It has the subject: **CN=SCC,OU=BTP Scenarios,O=Trust Community,C=DE**.
- The short-lived certificate has the subject **CN=P1234567890**, where P1234567890 is the platform user.

Tasks

[Prerequisites \[page 430\]](#)

[Configure an ABAP System to Trust the Cloud Connector's System Certificate \[page 431\]](#)

[Map Short-Lived Certificates to Users \[page 433\]](#)

[Access ICF Services \[page 433\]](#)

Prerequisites

- For identity propagation, mutual authentication (mTLS) is mandatory. To enable mTLS, you must configure a system certificate as described in this topic. Follow step 1 below to make sure the ABAP system trusts this certificate. You can use the connection check and the [Details](#) button to check if mTLS is working.

- The access control entry (see [Configure Access Control \(HTTP\) \[page 457\]](#)) must have specified the respective identity type to forward the identity correctly.

To perform the following steps, you must have the corresponding authorizations in the ABAP system for the transactions mentioned below (administrator role according to your specific authorization management) as well as an administrator user for the Cloud Connector.

Back to [Tasks \[page 430\]](#)

Back to [Example Data \[page 430\]](#)

1. Configure an ABAP System to Trust the Cloud Connector's System Certificate

This step includes two sub-steps:

[Configure the ABAP system to trust the Cloud Connector's system certificate \[page 431\]](#)

[Configure the Internet Communication Manager \(ICM\) to trust the system certificate for principal propagation \[page 431\]](#)

Configure the ABAP system to trust the Cloud Connector's system certificate:

1. Open the *Trust Manager* (transaction STRUST).
2. Double-click the *SSL-Server Standard* folder in the menu tree on the left.
3. In the displayed screen, click the *Import certificate* button.
4. In the dialog window, choose the certificate file representing the public key of the issuer of the system certificate, for example, in DER format. Typically, this is a CA certificate. If you decide to use a self-signed system certificate, it is the system certificate itself.
5. The details of this certificate are shown in the section above. Using the example certificate data, you would see **CN=MyCompany CA, O=Trust Community, C=DE** as subject.
6. If you are sure that you are importing the correct certificate, you can integrate the certificate into the certificate list by choosing the *Add to Certificate List* button.
7. Now, the CA certificate (**CN=MyCompany CA, O=Trust Community, C=DE**) is part of the certificate list.

Back to [Step \[page 431\]](#)

Configure the Internet Communication Manager (ICM) to trust the system certificate for identity propagation:

1. Open the *Profile Editor* (transaction RZ10).
2. Select the profile you want to edit, for example, the DEFAULT profile.
3. Select the *Extended maintenance* radio button and choose the *Change* button.
4. Create the following parameter: `icm/trusted_reverse_proxy_<x> = SUBJECT="<subject>", ISSUER="<issuer>"`.
 - Select a free index for **<x>**.
 - **<subject>** is the subject of the system certificate (example data: **CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE**).

- **<issuer>** is the issuer of the system certificate (example data: **CN=MyCompany CA, O=Trust Community, C=DE**).
- Example: `icm/trusted_reverse_proxy_2 = SUBJECT="CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE", ISSUER="CN=MyCompany CA, O=Trust Community, C=DE"`.

Note

If your ABAP system uses kernel 7.42 or lower, see SAP note [2052899](#) or set the following two parameters:

- `icm/HTTPS/trust_client_with_issuer`: this is the issuer of the system certificate (example data: **CN=MyCompany CA, O=Trust Community, C=DE**).
- `icm/HTTPS/trust_client_with_subject`: this is the subject of the system certificate (example data: **CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE**).

Caution

The ICM expects *blanks after the separating comma* for the issuer and subject elements. So, even if the Cloud Connector administration UI shows a string without blanks, for example: `CN=SCC,OU=BTP Scenarios,O=Trust Community,C=DE`, you must specify in ICM: `CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE`.

Caution

Make sure that `icm/HTTPS/verify_client` is set to either **1** (request certificate) or **2** (require certificate). If the parameter is set to **0**, trust cannot be established.

5. Save the profile.
6. Open the *ICM Monitor* (transaction `SMICM`) and restart the ICM by choosing **Administration > ICM > Exit Hard > Global**.
7. Verify that the two profile parameters have been taken over by ICM by choosing **Goto > Parameters > Display**.

Note

If you have an SAP Web Dispatcher installed in front of the ABAP system, trust must be added in its configuration files with the same parameters as for the ICM. Also, you must add the system certificate of the Cloud Connector to the trust list of the Web dispatcher Server PSE. For more information, see [Configure Identity Propagation via SAP Web Dispatcher \[page 435\]](#).

Caution

When using identity propagation with X.509 certificates, you cannot use the strict mode in certificate block management (transaction code: `CRCONFIG`) for the CRL checks within profile `SSL_SERVER`.

Back to [Step \[page 431\]](#)

Back to [Tasks \[page 430\]](#)

Back to [Example Data \[page 430\]](#)

2. Map Short-Lived Certificates to Users

For systems later than SAP NetWeaver 7.3 EHP1 (7.31), you can use rule-based certificate mapping, which is the recommended way to create the required user mappings. For more information, see [Rule-Based Mapping of Certificates \[page 441\]](#).

In older releases (for which this feature does not exist yet), you can do this manually in the system as described below, or use an identity management solution generating the mapping table for a more comfortable approach.

1. Open *Assignment of External ID to Users* (transaction `EXTID_DN`).
2. Switch to the edit mode.
3. Create a new entry. Specify the subject of the certificate as `External ID`. When using the example data, this is `CN=P1234567890`. In the *User* field, provide the appropriate ABAP user, for example `JOHNSMITH`.
4. Choose *Activate*.
5. Save the mapping.
6. Repeat the previous steps for all users that should be supported for the scenario.

Back to [Tasks \[page 430\]](#)

Back to [Example Data \[page 430\]](#)

3. Access ICF Services

To access the required ICF services for your scenario in the ABAP system, choose one of the following procedures:

For Cloud Connector version 2.15 or higher:

- To access ICF services via **certificate logon** (using the system certificate), select *Allow Principal Propagation*, choose the principal type **x.509 Certificate**, and select *System Certificate for Logon* in the corresponding system mapping.
This setting lets you use the system certificate for trust in case of principal propagation as well as for user authentication. For details, see [Configure Access Control \(HTTP\) \[page 457\]](#), step 7, 8 (*Procedure for Cloud Connector version 2.15 and higher*), and 9.
Additionally, make sure that all required ICF services allow *Logon Through SSL Certificate* as logon method.
- To access ICF services via the logon method **Basic Authentication** (logon with user/password) and identity propagation, select *Allow Principal Propagation*, choose the principal type **x.509 Certificate**, and do *not* select *System Certificate for Logon* in the corresponding system mapping.
This setting lets you use the system certificate for trust, but prevents its usage for user authentication. For details, see [Configure Access Control \(HTTP\) \[page 457\]](#), step 7, 8 (*Procedure for Cloud Connector version 2.15 and higher*), and 9.
Additionally, make sure that all required ICF services allow *Basic Authentication* and *Logon Through SSL Certificate* as logon method.
- If some of the ICF services require `Basic Authentication`, while others should be accessed via system certificate logon, perform these steps:

1. In the Cloud Connector system mapping, select *Allow Principal Propagation*, choose the principal type **X.509 Certificate**, and select *System Certificate for Logon* in the corresponding system mapping as described above.
2. In the ABAP system, choose transaction code SICF and go to *Maintain Services*.
3. Select the service that requires `Basic Authentication` as logon method.
4. Double-click the service and go to tab *Logon Data*.
5. Switch to *Alternative Logon Procedure* and ensure that the `Basic Authentication` logon procedure is listed before `Logon Through SSL Certificate`.

For Cloud Connector versions below 2.15:

- To access ICF services via certificate logon, choose the principal type **X.509 Certificate (general usage)** in the corresponding system mapping. This setting lets you use the system certificate for trust as well as for user authentication.
For details, see [Configure Access Control \(HTTP\) \[page 457\]](#), step 8 (*Procedure for Cloud Connector versions below 2.15*).
Additionally, make sure that all required ICF services allow `Logon Through SSL Certificate` as logon method.
- To access ICF services via the logon method `Basic Authentication` (logon with user/password) and principal propagation, choose the principal type **X.509 Certificate (strict usage)** in the corresponding system mapping. This setting lets you use the system certificate for trust, but prevents its usage for user authentication.
For details, see [Configure Access Control \(HTTP\) \[page 457\]](#), step 8 (*Procedure for Cloud Connector versions below 2.15*).
Additionally, make sure that all required ICF services allow `Basic Authentication` and `Logon Through SSL Certificate` as logon methods.
- If some of the ICF services require `Basic Authentication`, while others should be accessed via system certificate logon, perform these steps:
 1. In the Cloud Connector system mapping, choose the principal type **X.509 Certificate (general usage)** as described above.
 2. In the ABAP system, choose transaction code SICF and go to *Maintain Services*.
 3. Select the service that requires `Basic Authentication` as logon method.
 4. Double-click the service and go to tab *Logon Data*.
 5. Switch to *Alternative Logon Procedure* and ensure that the `Basic Authentication` logon procedure is listed before `Logon Through SSL Certificate`.

Note

If you are using SAP Web Dispatcher for communication, you must configure it to forward the SSL certificate to the ABAP backend system, see [Forward SSL Certificates for X.509 Authentication](#) (SAP Web Dispatcher documentation).

Back to [Tasks \[page 430\]](#)

Back to [Example Data \[page 430\]](#)

Related Information

[Configure Identity Propagation via SAP Web Dispatcher \[page 435\]](#)

[Rule-Based Mapping of Certificates \[page 441\]](#)

[Configure Subject Patterns for Principal Propagation \[page 443\]](#)

[Set Up Trust \[page 420\]](#)

[Principal Propagation \[page 168\]](#) (Cloud Foundry environment)

[Principal Propagation](#) (Neo environment)

1.2.2.3.1.3.1.1 Configure Identity Propagation via SAP Web Dispatcher

Set up a trust chain to use identity propagation to an ABAP System for HTTPS via SAP Web Dispatcher.

Note

The information in this section applies to both principal propagation and technical user propagation. These two types of user propagation are combined as *identity* propagation.

Concept

If you are using an intermediate SAP Web Dispatcher to connect to your ABAP backend system, you must set up a trust chain between the involved components Cloud Connector, SAP Web Dispatcher, and ABAP backend system.

Before configuring the ABAP system (see [Configure Identity Propagation for HTTPS \[page 430\]](#)), in a first step you must configure SAP Web Dispatcher to accept and forward user principals propagated from a cloud account to an ABAP backend.

To do this, follow the step-by-step instructions below.

Example Data

The following data and setup is used for this scenario:

- System certificate was issued by `CN=MyCompany CA, O=Trust Community, C=DE`
- It has the subject `CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE`

Tasks

- [Prerequisites \[page 436\]](#)
- [Configure the SAP Web Dispatcher to Trust the Cloud Connector's System Certificate \[page 436\]](#)
- [Next Steps \[page 438\]](#)

Prerequisites

- Your SAP Web Dispatcher version is 7.53 or higher. See SAP note [908097](#) for information on recommended SAP Web Dispatcher versions.
- We recommend that you use a standalone Web Dispatcher deployment. To learn about deployment options, see SAP note [3115889](#).
- Make sure your SAP Web Dispatcher supports SSL. See [Configure SAP Web Dispatcher to Support SSL](#).
- Ensure that TLS client certificates can be used for authentication in the backend system. See [How to Configure SAP Web Dispatcher to Forward SSL Certificates for X.509 Authentication](#) for step-by-step instructions.

Back to [Tasks \[page 436\]](#)

Back to [Concept \[page 435\]](#)

Configure SAP Web Dispatcher to Trust the Cloud Connector's System Certificate

To allow Cloud Connector client certificates for authentication in the backend system, perform the following two steps:

1. **Configure SAP Web Dispatcher to trust the Cloud Connector's system certificate:**
 1. To import the system certificate to SAP Web Dispatcher, open the SAP Web Dispatcher administration interface in your browser.

Note

The interface is usually configured on `/sap/wdisp/admin`.

2. In the menu, navigate to *SSL and Trust Configuration* and select *PSE Management*.
3. In the **Manage PSE** section, select `SAPSSLS.pse` from the drop-down list. By default, SAPSSLS.pse contains the server certificate and the list of trusted clients that SAP Web Dispatcher trusts as a server.
4. In the **Trusted Certificates** section, choose *Import Certificate*.
5. Enter the certificate as *base64-encoded* into the text box. The procedure to export your certificate in such a format is described in [Forward SSL Certificates for X.509 Authentication](#), step 1.

Note

Typically, this is a CA certificate. If you are using a self-signed system certificate, it's the system certificate itself.

6. Choose *Import*.

7. The certificate details are now shown in section **Trusted Certificates**.

2. Configure SAP Web Dispatcher to trust the Cloud Connector's system certificate for identity propagation:

- Create or edit the following parameter in SAP Web Dispatcher:
`icm/trusted_reverse_proxy_<x> = SUBJECT="<subject>", ISSUER="<issuer>"`
 - Select a free index for <x>.
 - <subject>: Subject of the system certificate (example data: CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE)
 - <issuer>: Issuer of the system certificate (example data: CN=MyCompany CA, O=Trust Community, C=DE)

Example: `icm/trusted_reverse_proxy_0 = SUBJECT="CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE", ISSUER="CN=MyCompany CA, O=Trust Community, C=DE"`

Caution

The ICM expects *blanks after the separating comma* for the issuer and subject elements. So, even if the Cloud Connector administration UI shows a string without blanks, for example: `CN=SCC,OU=BTP Scenarios,O=Trust Community,C=DE`, you must specify in ICM: `CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE`.

- **[Deprecated]** Create or edit the following two parameters in SAP Web Dispatcher:

Note

Use the parameters below (instead of `icm/trusted_reverse_proxy_<x>`) only if your kernel release does not yet support parameter `icm/trusted_reverse_proxy_<x>`.

- `icm/HTTPS/trust_client_with_issuer`: Issuer of the system certificate (example data: CN=MyCompany CA, O=Trust Community, C=DE)
- `icm/HTTPS/trust_client_with_subject`: Subject of the system certificate (example data: CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE)

Note

Make sure `icm/HTTPS/verify_client` is set to **1** (request certificate) or **2** (require certificate). If set to **0**, trust cannot be established. The default value is **1**, so it is OK if the parameter is not set at all.

Back to [Tasks \[page 436\]](#)

Back to [Concept \[page 435\]](#)

Next Steps

Now you can proceed with:

- Step 1 of the basic identity propagation setup for HTTPS, see [Configure an ABAP System to Trust the Cloud Connector's System Certificate \[page 431\]](#). However, when using SAP Web Dispatcher, the ABAP backend must trust the *SAP Web Dispatcher* instead of the *Cloud Connector*, see [Forward SSL Certificates for X.509 Authentication](#), step 2 for details.

Then perform the remaining steps of the basic identity propagation setup for HTTPS as described here:

- [Map Short-Lived Certificates to Users \[page 433\]](#)
- [Access ICF Services \[page 433\]](#)

Back to [Tasks \[page 436\]](#)

Back to [Concept \[page 435\]](#)

1.2.2.3.1.3.2 Configure Identity Propagation for RFC

Find step-by-step instructions to configure principal propagation to an ABAP server for RFC.

Configuring principal propagation for RFC requires an SNC (Secure Network Communications) connection. To enable SNC, you must configure the ABAP system and the Cloud Connector accordingly.

The following example provides step-by-step instructions for the SNC setup.

Note

It is important that you use the same SNC implementation on both communication sides. Contact the vendor of your SNC solution to check the compatibility rules.

Example Data

The following data and setup is used:

Note

The parameters provided in this example are based on an SNC implementation that uses the **SAP Cryptographic Library**. Other vendors' libraries may require different values.

- An SNC identity has been generated and installed on the Cloud Connector host. Generating this identity for the SAP Cryptographic Library is typically done using the tool `SAPGENPSE`. For more information, see [Configuring SNC for SAPCRYPTOLIB Using SAPGENPSE](#).

- The ABAP system is configured properly for SNC.

Note

For the latest system releases, you can use the SSO wizard to configure SNC (transaction code: SNCWIZARD). System prerequisites are described in SAP note [2015966](#).

- The Cloud Connector system identity's SNC name is **p:CN=SCC, OU=SAP CP Scenarios, O=Trust Community, C=DE**.
- The ABAP system's SNC identity name is **p:CN=SID, O=Trust Community, C=DE**. This value can typically be found in the ABAP system instance profile parameter `snc/identity/as` and hence is provided per application server.
- When using the SAP Cryptographic Library, the ABAP system's SNC identity and the Cloud Connector's system identity should be signed by the same CA for mutual authentication.
- The example short-lived certificate has the subject **CN=P1234567**, where **P1234567** is the SAP BTP application user.

Tasks

1. [Configure the ABAP System \[page 439\]](#)
2. [Map Short-Lived Certificates to Users \[page 440\]](#)
3. [Configure the Cloud Connector \[page 440\]](#)

1. Configure the ABAP System to Trust the Cloud Connector's System SNC identity

1. Open the *SNC Access Control List for Systems* (transaction `SNC0`).
2. As the Cloud Connector does not have a system ID, use an arbitrary value for `<System ID>` and enter it together with its SNC name: **p:CN=SCC, OU=SAP CP Scenarios, O=Trust Community, C=DE**.
3. Save the entry and choose the *Details* button.
4. In the next screen, activate the checkboxes for *Entry for RFC activated* and *Entry for certificate activated*.
5. Save your settings.

Back to [Tasks \[page 439\]](#)

Back to [Example Data \[page 438\]](#)

2. Map Short-Lived Certificates to Users

You can do this manually in the system as described below or use an identity management solution for a more comfortable approach. For example, for large numbers of users the rule-based certificate mapping is a good way to save time and effort. See [Rule-Based Certificate Mapping](#).

1. Open *Assignment of External ID to Users* (transaction `EXTID_DN`).
2. Switch to the edit mode.
3. Create a new entry. Specify the subject of the certificate as `EXTERNAL ID`. Using the example data, this is `CN=P1234567`. In the `<User>` field, provide an appropriate ABAP user, for example `JOHNDOE`.
4. Save the mapping.
5. Repeat the previous steps for all users that should be supported for the scenario.

Back to [Tasks \[page 439\]](#)

Back to [Example Data \[page 438\]](#)

3. Configure the Cloud Connector

[Prerequisites \[page 440\]](#)

[Set up the Cloud Connector to Use the SNC Implementation \[page 440\]](#)

[Create an RFC Hostname Mapping \[page 441\]](#)

Prerequisites

- The required security product for the SNC flavor that is used by your ABAP back-end systems, is installed on the Cloud Connector host.
- The Cloud Connector's system SNC identity is associated with the operating system user under which the Cloud Connector process is running.

Note

If you use SAP Cryptographic Library as SNC implementation, follow the steps described in [Initial Configuration \(RFC\) \[page 398\]](#). Additionally, SAP note [2642538](#) provides a good description to associate an SNC identity of SAP Cryptographic Library with a user running an external program that uses JCo. When using a different SNC offering, get in touch with the SNC library vendor for details.

Back to [Step \[page 440\]](#)

Set up the Cloud Connector to Use the SNC Implementation

1. In the Cloud Connector UI, choose *Configuration* from the main menu, select the *On Premise* tab, and go to the *SNC* section.
2. Provide the fully qualified name of the SNC library (the security product's shared library implementing the GSS API), the SNC name of the above system identity, and the desired quality of protection by choosing the *Edit* icon.
For more information, see [Initial Configuration \(RFC\) \[page 398\]](#).

Note

The example in [Initial Configuration \(RFC\) \[page 398\]](#) shows the library location if you use the SAP Secure Login Client as your SNC security product. In this case (as well as for some other security products), *SNC My Name* is optional, because the security product automatically uses the identity associated with the current operating system user under which the process is running, so you can leave that field empty. (Otherwise, in this example it should be filled with `p:CN=SCC, OU=SAP CP Scenarios, O=Trust Community, C=DE.`)

We recommend that you enter **Maximum Protection** for `<Quality of Protection>`, if your security solution supports it, as it provides the best protection.

3. Choose *Save* and *Close*.

Back to [Step \[page 440\]](#)

Create an RFC Hostname Mapping

1. In the *Access Control* section of the Cloud Connector, create a hostname mapping corresponding to the cloud-side RFC destination. See [Configure Access Control \(RFC\) \[page 465\]](#).
2. Make sure you choose **RFC SNC** as `<Protocol>` and **ABAP System** as `<Back-end Type>`. In the `<SNC Partner Name>` field, enter the ABAP system's SNC identity name, for example, `p:CN=SID, O=Trust Community, C=DE.`
3. Save your mapping.

Back to [Step \[page 440\]](#)

Back to [Tasks \[page 439\]](#)

Back to [Example Data \[page 438\]](#)

Related Information

[Secure Network Communications \(SNC\)](#)

[Using the SAP Cryptographic Library for SNC](#)

[Rule-Based Mapping of Certificates \[page 441\]](#)

[Principal Propagation \[page 168\]](#) (Cloud Foundry environment)

[Principal Propagation](#) (Neo environment)

1.2.2.3.1.3.3 Rule-Based Mapping of Certificates

Learn how to efficiently map short-lived certificates to users in the ABAP server.

Note

The information in this section applies to both principal propagation and technical user propagation.

To perform rule-based mapping of certificates in the ABAP server, proceed as follows:

1. Enter a dynamic parameter using transaction RZ11.
 1. Enter the `login/certificate_mapping_rulebased` parameter.
 2. Choose the *Change Value* button.
 3. Enter the value **1**.
 4. Save the value.

Note

If dynamic parameters are disabled, enter the value using transaction RZ10 and restart the whole ABAP system.

2. Configure rule-based mapping
 1. To create a sample certificate with the Cloud Connector, logon to the Cloud Connector UI and from the main menu, choose *Configuration*. In the *System Certificate* section, enter a sample CN name and download the sample certificate to the *Downloads* folder of your machine.
 2. To import the sample certificate into the ABAP server, choose transaction CERTRULE and select *Import certificate*.

Note

To access transaction CERTRULE, you need the corresponding authorizations (see: [Assign Authorization Objects for Rule-based Mapping \[page 443\]](#)).

3. To create explicit rule mappings, choose the *Rule* button.
4. Choose *Save*.

Note

When you save the changes and return to transaction CERTRULE, the sample certificate which you imported in Step 2b will not be saved. This is just a sample editor view to see the sample certificates and mappings.

Related Information

[Rule-Based Certificate Mapping](#)

1.2.2.3.1.3.3.1 Assign Authorization Objects for Rule-based Mapping

Assign authorizations to access transaction CERTRULE.

To access transaction CERTRULE, you need the following authorizations:

- CC control center: System administration (S_RZL_ADM)
 - `Activity 03` grants display authorizations.
 - `Activity 01` grants change authorizations.
- User Master Maintenance: User Groups (S_USER_GRP)
 - `Activity 03` grants display authorizations.
 - `Activity 02` grants change authorizations.
 - `Class`: enter the names of user groups for which the administrator can maintain explicit mappings.

To assign these authorization objects, proceed as follows:

1. Create a *Single Role* using transaction PFCG.
2. To add the authorization objects `S_RZL_ADM` and `S_USER_GRP`, go to the *Authorizations* tab, choose *Change Authorization data* and select the *Manually* button .
3. To generate the profile, choose *Generate* and save the changes.
4. In the *User* tab, enter the user who should execute the transaction CERTRULE.
5. To match the generated profile to the users, choose *User comparison* .

1.2.2.3.1.4 Configure Subject Patterns for Principal Propagation

Define patterns identifying the user for the subject of a generated short-lived X.509 certificate.

Note

The information in this section applies to both principal propagation and technical user propagation.

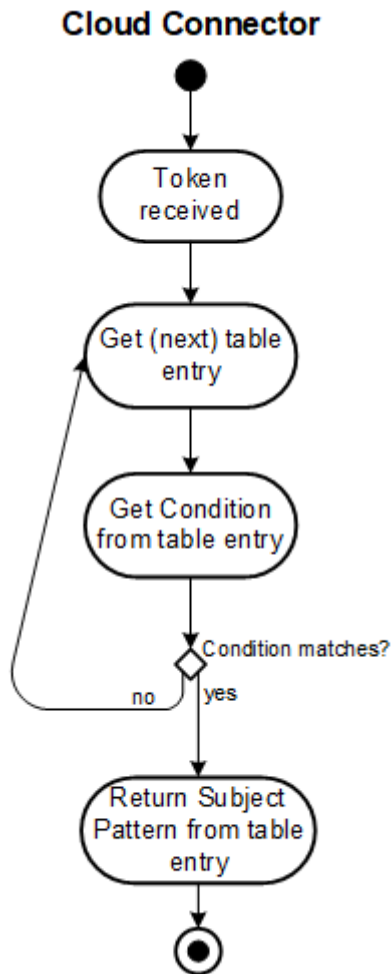
Using this configuration option, you can define different patterns identifying the user for the subject of the generated short-lived X.509 certificate, based on a specified condition. You can also specify the validity period and expiration tolerance.

Configure Subject Patterns

To configure a subject pattern, choose ► *Configuration* ► *On Premise* ► *Principal Propagation* ►. In the table shown, you can add or modify patterns.

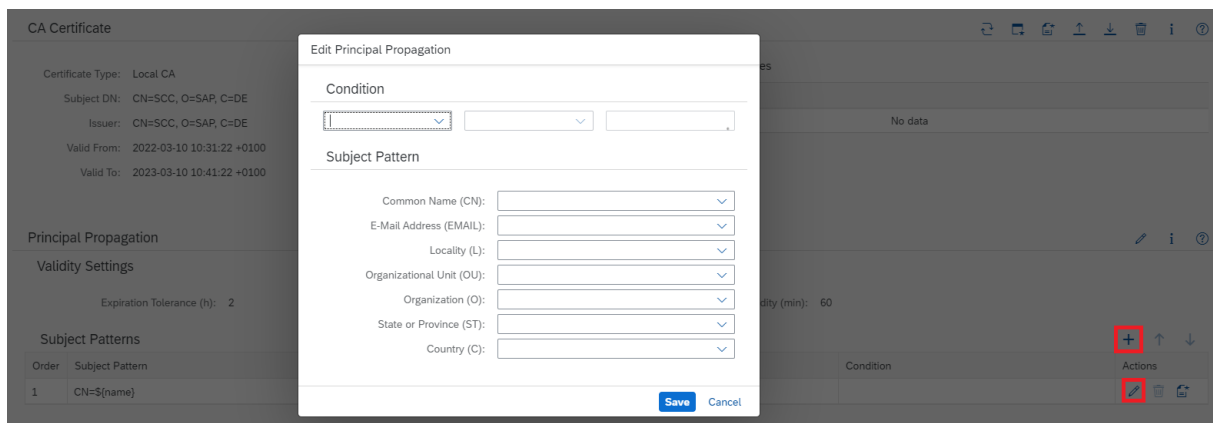
Subject Patterns			+	↑	↓
Order	Subject Pattern	Condition	Actions		
1	CN=\${mail}	\$(mail) exists			
2	CN=\${name}				

This table represents an ordered list containing entries that have a specified condition, and the respective subject pattern. You can change the order for an entry by choosing the arrow buttons. The workflow in the Cloud Connector looks like this:



The last entry in the table is the default one having no condition (that is, it always matches as fallback). This entry can not be moved or deleted.

To modify or add table entries, choose the *Edit* or *Add* icon:



Specify a Condition

Use either of the following procedures to specify a condition based on the attributes of incoming tokens from cloud side:

- Enter the values in the subject pattern fields manually.
- Use the selection menu of the corresponding field to enter a predefined variable.

Using the selection menu, you can assign values for the following parameters:

- `${user_type}`
- `${name}`
- `${mail}`
- `${email}`
- `${display_name}`
- `${login_name}`

Operators

In a next step, choose an operator:

- *exists*: This attribute must be present in the incoming token.
- *does not exist*: This attribute must not be present in the incoming token.
- *is*: This attribute must be present and equals to the value that has been entered in the third field afterwards.
- *is not*: This attribute must be present and is not equal to the value that has been entered in the third field afterwards.

📌 Note

For the condition `${user_type}`, you can only switch between **Technical** or **Business**. The latter refers to the "classical" propagation of business user information, whereas **Technical** is the propagation of a technical user.

Subject Pattern Details

Use either of the following procedures to define the subject's distinguished name (DN), for which the certificate will be issued:

- Enter the values in the subject pattern fields manually.
- Use the selection menu of the corresponding field to enter a predefined variable.

Using the selection menu, you can assign values for the following parameters:

- `${name}`
- `${mail}`
- `${display_name}`
- `${login_name}`

📌 Note

If the token provided by the Identity Provider contains additional values that are stored in attributes with different names, but you still want to use it for the subject pattern, you can edit the variable name to place the corresponding attribute value in the subject accordingly. For example, provide `${email}`, if a SAML assertion uses `email` instead of providing `mail`, or `${user_uuid}` if the attribute `user_uuid` representing the global user ID is contained in the assertion.

When using a subaccount in the **Cloud Foundry** environment: The Cloud Connector also offers direct access to custom variables injected in the JWT (JSON Web token) by SAP BTP *Authorization & Trust Management* that were taken over from a SAML assertion.

The values for these variables are provided by the trusted identity provider in the token which is passed to the Cloud Connector and specifies the user that has logged on to the cloud application.

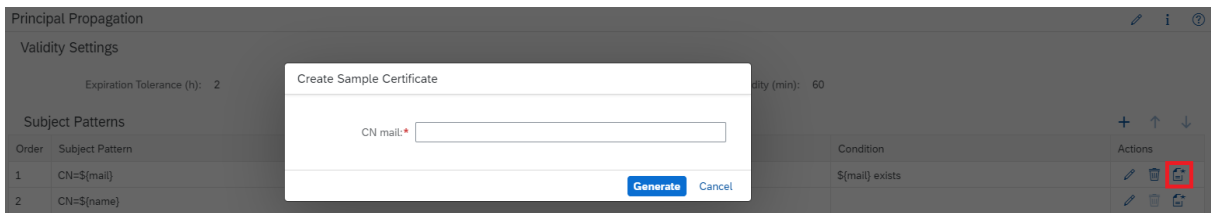
By default, the following attributes are provided:

- `<CN>`: (common name) – the name of the certificate owner
- `<EMAIL>`: (e-mail address) - the e-mail address of the certificate owner
- `<L>`: (locality) – the certificate owner's location
- `<O>`: (organization) – the certificate owner's organization or company
- `<OU>`: (name of organizational unit) – the organizational unit to which the certificate owner belongs
- `<ST>`: (state of residence) – the state in which the certificate issuer resides
- `<C>`: (country of residence) – the country in which the certificate owner resides
- `<Expiration Tolerance (h)>`: The length of time in hours, that an application can use a principal issued for a user after the token from cloud side has expired.

- `<Certificate Validity (min)>`: The length of time in minutes, that a certificate generated for principal propagation can authenticate against the back end. You can reuse a previously generated certificate to improve performance.

Sample Certificate

By choosing [Generate Sample Certificate](#) you can create a sample certificate that looks like one of the short-lived certificates created at runtime. You can use this certificate to, for example, generate user mapping rules in the target system, via transaction CERTRULE in an ABAP system. If your subject pattern contains variable fields, a wizard lets you provide meaningful values for each of them and eventually you can save the sample certificate in DER format.



Validity Settings

You can change the validity settings by choosing the [Edit](#) button.

Related Information

[Server Certificate Authentication \[page 89\]](#)

1.2.2.3.1.5 Configure a Secure Login Server

Configuration steps for Java SLS support.

ⓘ Note

The information in this section applies to both principal propagation and technical user propagation.

📌 Note

The Secure Login Server mainstream maintenance ends on December 31, 2027.

Content

[Overview \[page 448\]](#)

[Requirements \[page 449\]](#)

[Implementation \[page 449\]](#)

Overview

The Cloud Connector can use on-the-fly generated X.509 user certificates to log in to on-premise systems if the external user session is authenticated (for example by means of SAML). If you do not want to use the built-in certification authority (CA) functionality of the Cloud Connector (for example because of security considerations), you can connect SAP SSO 2.0 Secure Login Server (SLS) or higher.

📌 Note

Make sure you use a version that is still supported, which is currently at least SAP SSO 3.0 Secure Login Server.

SLS is a Java application running on AS JAVA 7.20 or higher, which provides interfaces for certificate enrollment.

SLS supports the following formats:

- HTTPS
- REST
- JSON
- PKCS#10/PKCS#7

📌 Note

Any enrollment requires a successful user or client authentication, which can be a single, multiple or even a multi factor authentication.

The following schemes are supported:

- LDAP/ADS
- RADIUS
- SAP SSO OTP

- ABAP RFC
- Kerberos/SPNego
- X.509 TLS Client Authentication

SLS lets you define arbitrary enrollment profiles, each with a unique profile UID in its URL, and with a configurable authentication and certificate generation.

Back to [Content \[page 448\]](#)

Requirements

For user certification, SLS must provide a profile that adheres to the following:

- Cloud Connector client authentication by its X.509 system certificate
- Cloud Connector system certificate and SLS may live in different PKIs
- Cloud Connector hands over the full user's certificate subject name

With SAP SSO 2.0 SP06, SLS provides the following required features:

- TLS Client Authentication-based enrollment with `SecureLoginModuleUserDelegationWithSSL` (available since SP04)
- multi-PKI support is implemented by all standard components of Application Server (AS) JAVA, AS ABAP, HANA, by importing trusted root CA certificates
- SLS allows `PKCS10:SUBJECT` in a profile's certificate configuration (SP06)

Back to [Content \[page 448\]](#)

Implementation

INSTALLATION

Follow the standard installation procedures for SLS. This includes the initial setup of a PKI (public key infrastructure).

Note

SLS allows you to set up one or more own PKIs with Root CA, User CA, and so on. You can also import CAs as PKCS#12 file or use a hardware security module (HSM) as "External User CA".

Note

You should only use HTTPS connections for any communication with SLS. AS JAVA / ICM supports TLS, and the default configuration comes with a self-signed sever certificate. You may use SLS to replace this certificate by a PKI certificate.

CONFIGURATION

SSL Ports

1. Open the NetWeaver Administrator, choose **Configuration > SSL** and define a new port with **Client Authentication Mode = REQUIRED**.

Note

You may also define another port with **Client Authentication Mode = Do not request** if you did not do so yet.

2. Import the root CA of the PKI that issued your Cloud Connector system certificate.
3. Save the configuration and restart the Internet Communication Manager (ICM).

Authentication Policy

1. Open the NetWeaver Administrator (NWA, <https://<host:port>/nwa>).
2. From the top-level menu, choose **Configuration > Authentication and Single Sign-On**.
3. In the *Policy Configuration* table, switch to **Type = Custom**.
4. Choose **Add** to create a new policy and assign it a name, for example, **SecureLoginCloudConnector**.
5. Open **Edit** mode.
6. In the *Details* section of the authentication configuration, choose **Authentication Stack > Login Modules** and add **SecureLoginModuleUserDelegationWithSSL**.
7. In `<Rule1.subjectName>` and `<Rule1.issuerName>`, enter the respective certificate names of your Cloud Connector system certificate.
8. In the *Details* section of the authentication configuration, choose **Properties** and add the property `UserNameMapping` with value **VirtualUser**.
9. Save the policy.

Client Authentication Profile

1. Open the SLS Administration Console (SLAC, <https://host:port/slac>).
2. From the top-level menu, choose **Profile Management > Authentication Profiles**.
3. Create a new profile with **Client Type = Secure Login Client** and assign it a name, for example, **Cloud Connector User Certificates**.
4. Choose **User Authentication > Use Policy Configuration** and select **Policy Configuration Name = SecureLoginCloudConnector**.
5. Edit all required fields in the wizard according to your requirements.
6. In tab *Authentication Configuration*, check the box **Virtual User**.
7. Save your entries.
8. Select the new profile and open **Edit** mode.
9. Choose **Certificate Configuration > Certificate Name and Alternative Names** and set **Appendix Subject Name = (PKCS10:SUBJECT)**.
10. Leave all other fields in *Certificate Name and Alternative Names* empty.
11. On the *Enrollment Configuration* page, make sure that the `<Enrollment URL>` has the correct value, otherwise edit and fix it:
 1. full DNS name
 2. port with TLS Client Authentication (see port number in NWA SSL Configuration).
12. Save your entries.

User Profile Group

1. Open the SLS Administration Console (SLAC, <https://host:port/slac>).
2. Go to the top level menu and choose ► [Profile Management](#) ► [User Profile Groups](#) ►.
3. Create a new profile group, make sure the <Policy URL> has the correct value:
 1. Full DNS name
 2. Port *without* TLS Client Authentication (see port number in NWA SSL Configuration).
4. In tab [Profiles](#), add the profile `Cloud Connector User Certificates`.
5. Save your entries.

Root CA Certificate

1. Open SLS Administration Console (SLAC, <https://host:port/slac>).
2. Go to the top level menu and choose [Certificate Management](#).
3. Select the Root CA certificate you are using in your profile.
4. Choose ► [Export entry](#) ► [X.509 Certificate](#) ► and download the certificate file.

Cloud Connector

Follow the standard installation procedure of the Cloud Connector and configure SLS support as explained in [Configuration of a CA hosted by a Secure Login Server](#):

1. Enter the policy URL that points to the SLS user profile group.
2. Select the profile, for example, `Cloud Connector User Certificates`.
3. Import the Root CA certificate of SLS into the Cloud Connector's [Trust Store](#).

On-Premise Target Systems

Follow the standard configuration procedure for Cloud Connector support in the corresponding target system and configure SLS support.

To do so, import the Root CA certificate of SLS into the system's truststore:

- **AS ABAP:** choose transaction STRUST and follow the steps in [Maintaining the SSL Server PSE's Certificate List](#).
- **AS Java:** open the Netweaver Administrator and follow the steps described in [Configuring the SSL Key Pair and Trusted X.509 Certificates](#).

Back to [Content \[page 448\]](#)

1.2.2.3.1.6 Configure Kerberos

Context

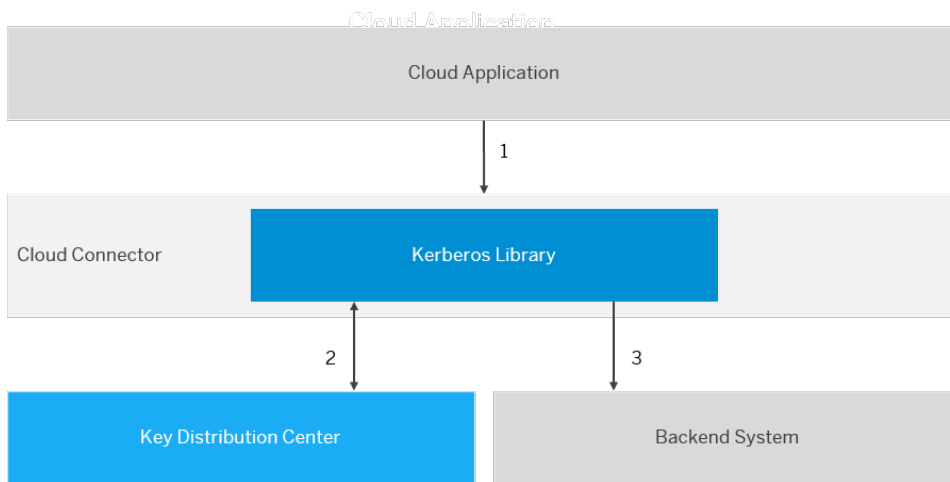
The Cloud Connector allows you to propagate users authenticated in SAP BTP via Kerberos against backend systems. It uses the **Service For User and Constrained Delegation** protocol extension of Kerberos.

Note

This feature is not supported for ABAP backend systems. In this case, you can use the certificate-based principal propagation, see [Configure a CA Certificate \[page 424\]](#).

The Key Distribution Center (KDC) is used for exchanging messages in order to retrieve Kerberos tokens for a certain user and backend system.

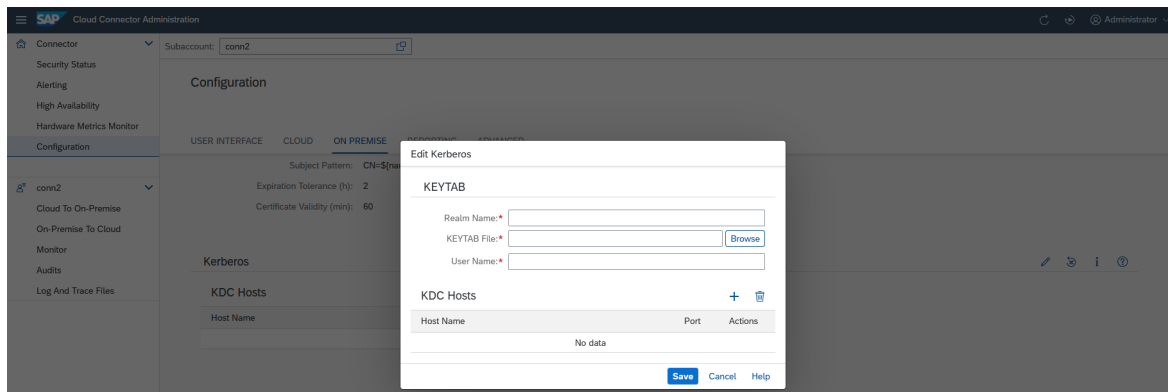
For more information, see [Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol](#).



1. An SAP BTP application calls a backend system via the Cloud Connector.
2. The Cloud Connector calls the KDC to obtain a Kerberos token for the user propagated from the Cloud Connector.
3. The obtained Kerberos token is sent as a credential to the backend system.

Procedure

1. Choose *Configuration* from the main menu.
2. From the *Kerberos* section of the *On Premise* tab, choose *Edit*.



3. Enter the name of your Kerberos realm.
4. Upload a KEYTAB file that contains the secret keys of your service user. The KEYTAB file should contain the `rc4-hmac` key for your user.
5. Enter the name of the service user to be used for communication with the KDC. This user should be allowed to request Kerberos tokens for other users for the backend systems that you are going to access.
6. In the `<KDC Hosts>` field (press `Add` to display the field), enter the host name of your KDC using the format `<host>:<port>`. The port is optional; if you leave it empty, the default, 88, is used.
7. Choose `Save`.

Example

You have a backend system protected with SPNego authentication in your corporate network. You want to call it from a cloud application while preserving the identity of a cloud-authenticated user.

Define the following:

- A connectivity destination in SAP BTP, with `ProxyType = OnPremise`.
- A system mapping made in the Cloud Connector. (Choose `Cloud to On Premise` from your subaccount menu, Go to tab `Access Control` `Add`, and for `Principal Type`, select `Kerberos`.)
- Kerberos configuration in the Cloud Connector, where the service user is allowed to delegate calls for your backend host service.

Result:

When you now call a backend system, the Cloud Connector obtains an SPNego token from your KDC for the cloud-authenticated user. This token is sent along with the request to the back end, so that it can authenticate the user and the identity to be preserved.

Related Information

[Set Up Trust \[page 420\]](#)

1.2.2.3.1.7 Configuring Identity Propagation to SAP NetWeaver AS for Java

Find step-by-step instructions on how to set up an application server for Java (AS Java) to enable identity propagation for HTTPS.

Note

The information in this section applies to both principal propagation and technical user propagation. These two types of user propagation are combined as *identity* propagation.

Prerequisites

To perform the following steps, you must have the corresponding administrator authorizations in AS Java (SAP NetWeaver Administrator) as well as an administrator user for the Cloud Connector.

Configure AS Java to Trust the Cloud Connector's System Certificate

Procedure

1. Go to ► [SAP NetWeaver Administrator](#) ► [Certificates and Keys](#) ► and import the Cloud Connector's system certificate into the *Trusted CAs* keystore view. See [Importing Certificate and Key From the File System](#).
2. Configure the Internet Communication Manager (ICM) to trust the system certificate for identity propagation.
 - a. Add a new SSL access point. See [Adding New SSL Access Points](#).
 - b. Generate a certificate signing request and send it to the CA of your choice. See [Configuration of the AS Java Keystore Views for SSL](#).
 - c. Import the certificates and save the configuration.

Import the certificate signing response, the root X.509 certificate of the trusted CA, and the Cloud Connector's system certificate into the new SSL access point from step 2a. Save the configuration and restart the ICM. See [Configuring the SSL Key Pair and Trusted X.509 Certificates](#).

- d. Make sure the ICM trusts the system certificate for identity propagation. For more information, see [Using Client Certificates via an Intermediary Server](#).
- e. Restart the ICM and test the SSL connection. For more information, see [Testing the SSL Connection](#).

Define Rules for User Mapping

Procedure

1. Add the *ClientCertLoginModule* to the policy configuration that the Cloud Connector connects to. See [Configuring the Login Module on the AS Java](#).
2. Define the rules to map users authenticated with their certificate to users that exist in the User Management Engine. See [Using Rules for User Mapping in Client Certificate Login Module](#).
 - To map the user ID of the certificate's subject name field to users, see [Using Rules Based on Client Certificate Subject Names](#).
 - To map the user ID based on rules for the certificate V3 extension *SubjectAlternativeName*, see [Using Rules Based on Client Certificate V3 Extensions](#).
 - To use client certificate filters, see [Defining Rules for Filtering Client Certificates](#).

Related Information

[Configuring Transport Layer Security on SAP NetWeaver AS for Java Using X.509 Client Certificates](#)
[Configure Identity Propagation for HTTPS \[page 430\]](#)

1.2.2.3.2 Configuring Technical User Propagation

Use technical user propagation to provide access of technical users to on-premise systems.

Using technical user propagation, you can forward the identity of technical users that are identified using an OAuth client and act on behalf of this identity in an on-premise system.

Note

Technical user propagation is possible only in the Cloud Foundry environment.

Tasks in this section:

Task	Description
Set Up Trust [page 420]	Configure a trusted relationship in the Cloud Connector to support technical user propagation. Technical user propagation lets you forward a technical user in the cloud to the internal system without requesting a password.
Configure a CA Certificate [page 424]	Install and configure an X.509 certificate to enable support for technical user propagation.
Configuring Identity Propagation to an ABAP System [page 429]	Learn more about the different types of configuring and supporting technical user propagation for a particular AS ABAP.
Configure Subject Patterns for Principal Propagation [page 443]	Define a pattern identifying the user for the subject of the generated short-lived X.509 certificate, as well as its validity period.
Configure a Secure Login Server [page 447]	Configuration steps for Java Secure Login Server (SLS) support.
Configure Kerberos [page 451]	The Cloud Connector lets you propagate users authenticated in SAP BTP via Kerberos against backend systems. It uses the Service For User and Constrained Delegation protocol extension of Kerberos.
Configuring Identity Propagation to SAP NetWeaver AS for Java [page 453]	Find step-by-step instructions on how to configure technical user propagation to an application server Java (AS Java).

Related Information

[OAuth Technical User Propagation Authentication \[page 145\]](#)

1.2.2.4 Configure Access Control

Specify the backend systems that can be accessed by your cloud applications.

To allow your cloud applications to access a certain backend system on the intranet, you must specify this system in the Cloud Connector. The procedure is specific to the protocol that you are using for communication.

Find the detailed configuration steps for each communication protocol here:

[Configure Access Control \(HTTP\) \[page 457\]](#)

[Configure Access Control \(RFC\) \[page 465\]](#)

[Configure Access Control \(LDAP\) \[page 472\]](#)

[Configure Access Control \(TCP\) \[page 474\]](#)

Copy Access Control Settings

When you add new subaccounts, you can copy the complete access control settings from another subaccount on the same Cloud Connector. You can also do it any time later by using the import/export mechanism provided by the Cloud Connector.

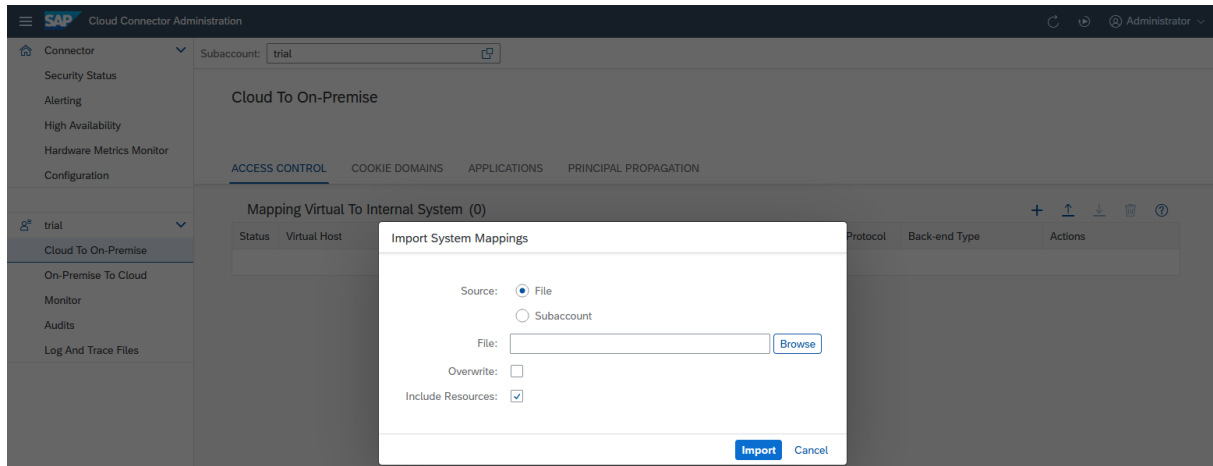
Export Access Control Settings

1. From your subaccount menu, choose *Cloud To On-Premise* and select the tab *Access Control*.
2. To store the current settings in a ZIP file, choose *Download* icon in the upper-right corner.
3. You can later import this file into a different Cloud Connector.

Import Access Control Settings

There are two locations from which you can import access control settings:

- A file that was previously exported from a Cloud Connector
- A different subaccount on the same Cloud Connector



Two additional options define the behavior of the import:

- **Overwrite:** Select this checkbox if you want to replace existing system mappings with imported ones. Do not select this checkbox if you want to keep existing mappings and only import the ones that are not yet available (default).

Note

A system mapping is uniquely identified by the combination of virtual host and port.

- **Include Resources:** When this checkbox is selected (default), the resources that belong to an imported system are also imported. Otherwise no resources are imported, that is, imported system mappings do not expose any resources.

Related Information

- [Configure Access Control \(HTTP\) \[page 457\]](#)
- [Configure Access Control \(RFC\) \[page 465\]](#)
- [Configure Access Control \(LDAP\) \[page 472\]](#)
- [Configure Access Control \(TCP\) \[page 474\]](#)
- [Configure Accessible Resources \[page 478\]](#)
- [Configure Domain Mappings for Cookies \[page 618\]](#)

1.2.2.4.1 Configure Access Control (HTTP)

Specify the backend systems that can be accessed by your cloud applications using HTTP.

To allow your cloud applications to access a certain backend system on the intranet via HTTP, you must specify this system in the Cloud Connector.

Note

Make sure that also redirect locations are configured as internal hosts.

If the target server responds with a redirect HTTP status code (30x), the cloud-side HTTP client usually sends the redirect over the Cloud Connector as well. The Cloud Connector runtime then performs a reverse lookup to rewrite the location header that indicates where to route the redirected request.

If the redirect location is ambiguous (that is, several mappings point to the same internal host and port), the first one found is used. If none is found, the location header stays untouched.

Tasks

[Expose Intranet Systems \[page 458\]](#)

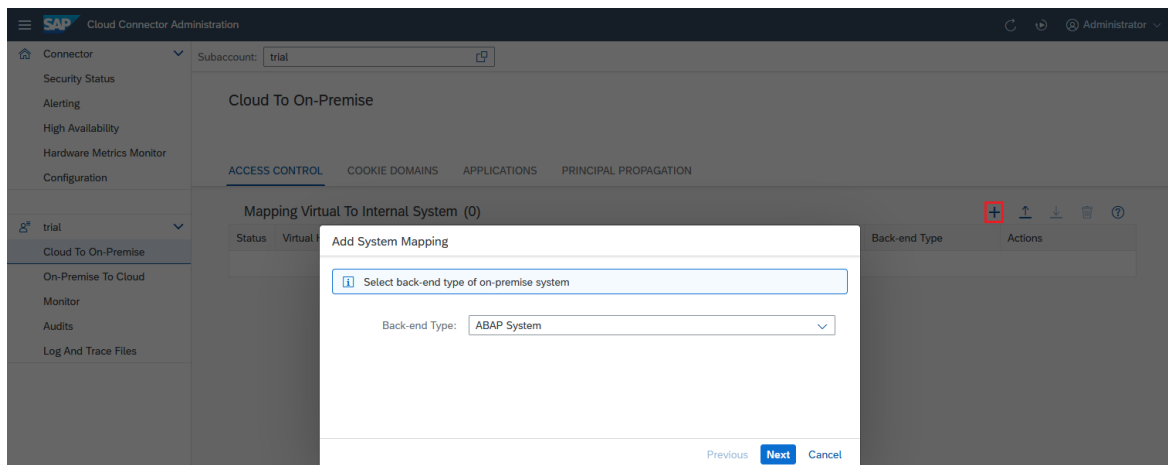
[Limit the Accessible Services for HTTP\(S\) \[page 464\]](#)

[Activate or Suspend Resources \[page 464\]](#)

Expose Intranet Systems

Insert a new entry in the Cloud Connector access control management:

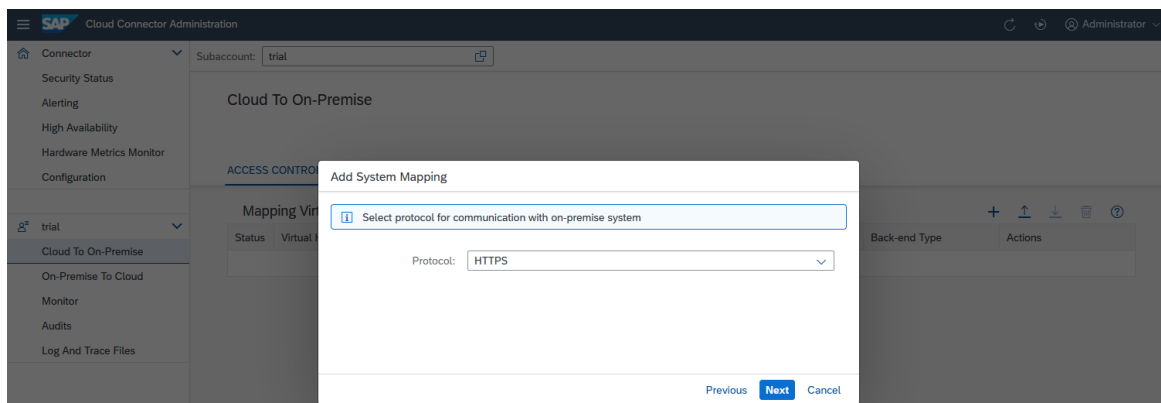
1. Choose *Cloud To On Premise* from your *Subaccount* menu.
2. Choose *Add*. A wizard will open and ask for the required values.
3. *Backend Type*: Select the description that matches best the addressed backend system.
When you are done, choose *Next*.



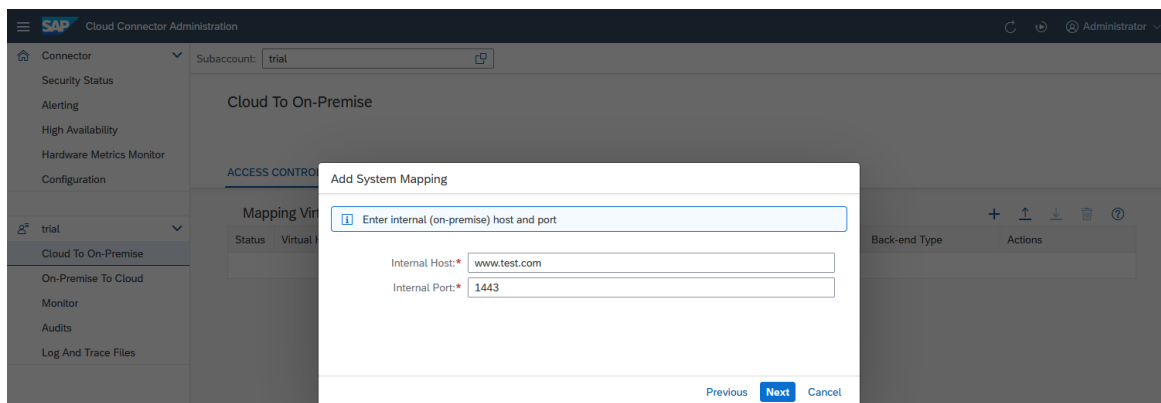
4. *Protocol*: This field allows you to decide whether the Cloud Connector should use **HTTP** or **HTTPS** for the connection to the backend system. Note that this is completely independent from the setting on cloud side. Thus, even if the HTTP destination on cloud side specifies "http://" in its URL, you can select **HTTPS**. This way, you are ensured that the entire connection from the cloud application to the actual backend system (provided through the TLS tunnel) is TLS-encrypted. The only prerequisite is that the

backend system supports HTTPS on that port. For more information, see [Initial Configuration \(HTTP\) \[page 396\]](#).

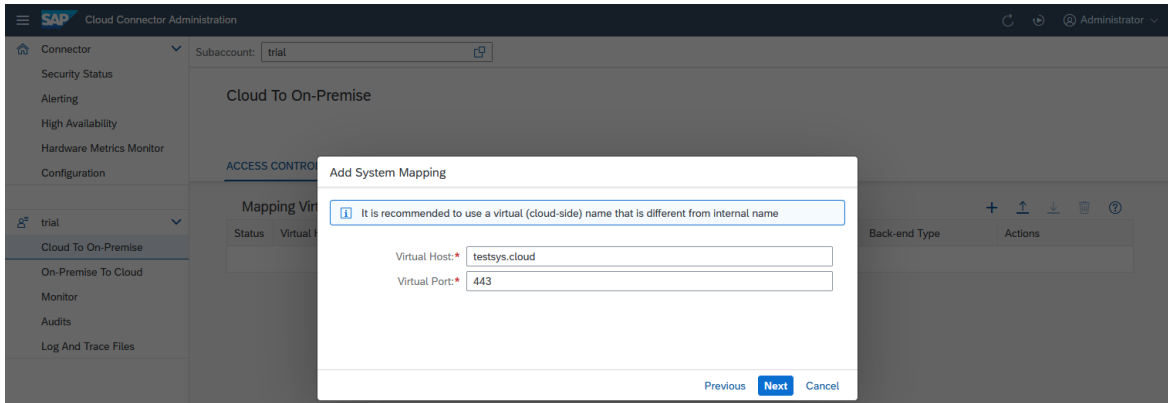
- If you specify HTTPS and there is a "system certificate" imported in the Cloud Connector, the latter attempts to use that certificate for performing a client-certificate-based logon to the backend system.
- If there is no system certificate imported, the Cloud Connector opens an HTTPS connection without client certificate.



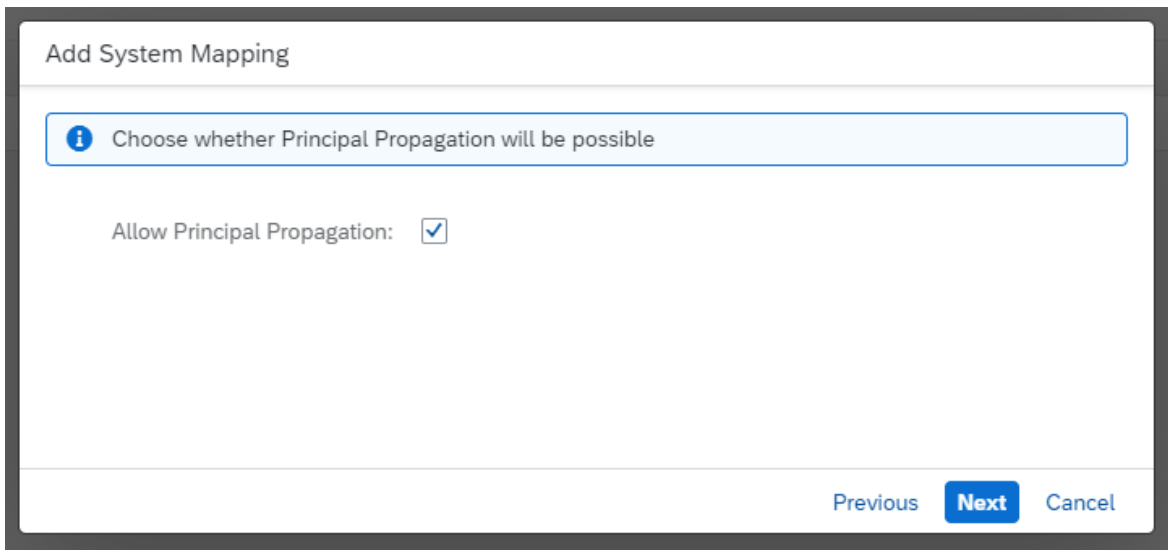
5. *Internal Host* and *Internal Port* specify the actual host and port under which the target system can be reached within the intranet. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector without any proxy. Cloud Connector will try to forward the request to the network address specified by the internal host and port, so this address needs to be real.



6. *Virtual Host* identifies the host name exactly as specified in the `<URL>` property of the HTTP destination configuration in SAP BTP. See: [Create HTTP Destinations \[page 63\]](#) (Cloud Foundry environment)
- The virtual host can be a fake name and does not need to exist. The *Virtual Port* allows you to distinguish between different entry points of your backend system, for example, **HTTP/80** and **HTTPS/443**, and to have different sets of access control settings for them. For example, some non-critical resources may be accessed by HTTP, while some other critical resources are to be called using HTTPS only. The fields are prepopulated with the values of the *Internal Host* and *Internal Port*. If you don't modify them, you must provide your internal host and port also in the cloud-side destination configuration or in the URL used for your favorite HTTP client.

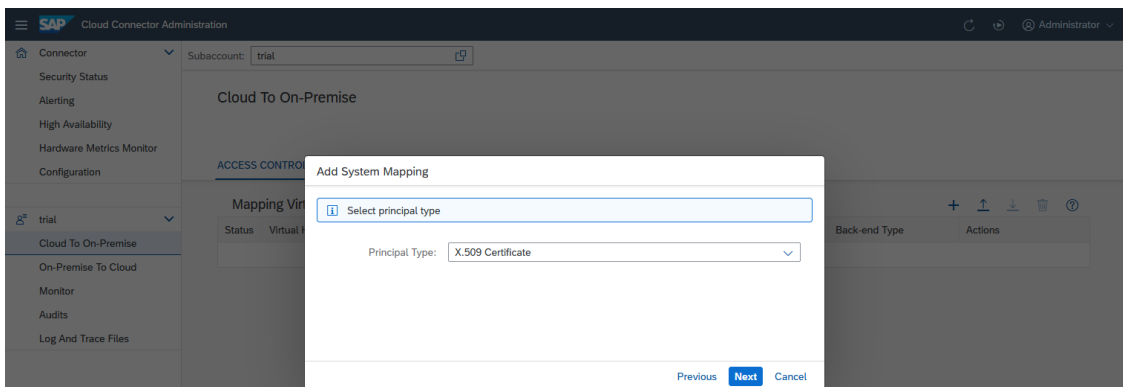


7. *Allow Principal Propagation* (available as of Cloud Connector 2.15): Defines if any kind of principal propagation should be allowed over this mapping. If not selected, go to step 9.



8. *Principal Type*

- *Procedure for Cloud Connector version 2.15 and higher:*
 Defines what kind of principal is sent to the backend within the HTTP request. For the principal type **X.509 Certificate** (if a principal is sent from the cloud side), the principal is injected as an HTTP header (SSL_CLIENT_CERT) and forwarded to the backend. There, depending on the backed configuration, it can be used for authentication.



- *Procedure for Cloud Connector versions below 2.15:*

You can use two different variants of an X.509 certificate to define the principal type that is sent to the backend within the HTTP request: **X.509 Certificate (General Usage)** or **X.509 Certificate (Strict Usage)**. The latter was introduced with Cloud Connector 2.11.

If a principal is sent from the cloud side, it is injected in both cases as an HTTP header (`SSL_CLIENT_CERT`) and forwarded to the backend. If the backend is configured correctly for principal propagation, this certificate can be used for authentication.

However, if the cloud side does not send a principal, the variants behave differently:

- *General Usage* (as well as `<Principal Type> = None`) allows to alternatively use the *system* certificate for the *TLS handshake* (actually used for trust) also for authentication.
- *Strict Usage* does not allow this. In this case, another authentication type (specified in an additional header) is used instead, for example, basic authentication.

This setting also applies to HTTP authentication types other than principal propagation.

Note

The recommended variant is **X.509 Certificate (Strict Usage)** as this lets you use principal propagation and, for example, basic authentication over the same access control entry, regardless of the logon order settings in the target system.

To prevent the use of principal propagation to the target system, choose **None** as `<Principal Type>`. In this case, no principal is injected.

For more information on principal propagation, see [Configuring Principal Propagation \[page 420\]](#).

9. *System Certificate for Logon* (available as of Cloud Connector 2.15): Specifies if the Cloud Connector's system certificate should be used for authentication at the backend, if
 1. No principal is received, or
 2. Principal propagation is not allowed over this mapping at all.

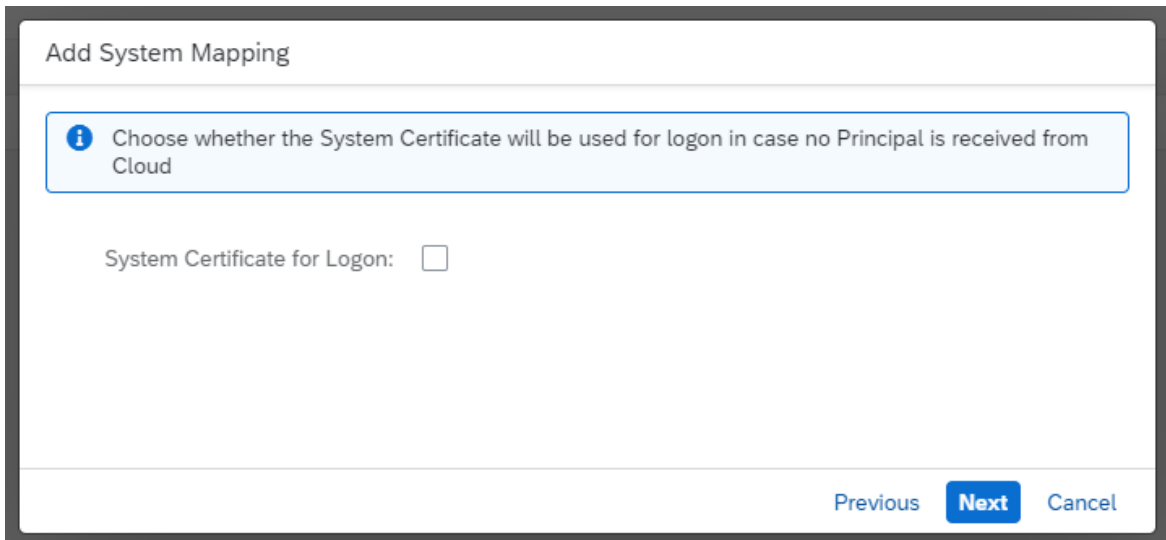
If *activated*, the system certificate of the TLS handshake used for trust is also used for authentication.

If *not activated*, an additional HTTP header (`SSL_CLIENT_CERT`) is sent. It indicates to the target system that the system certificate used for trust must not be used for authentication.

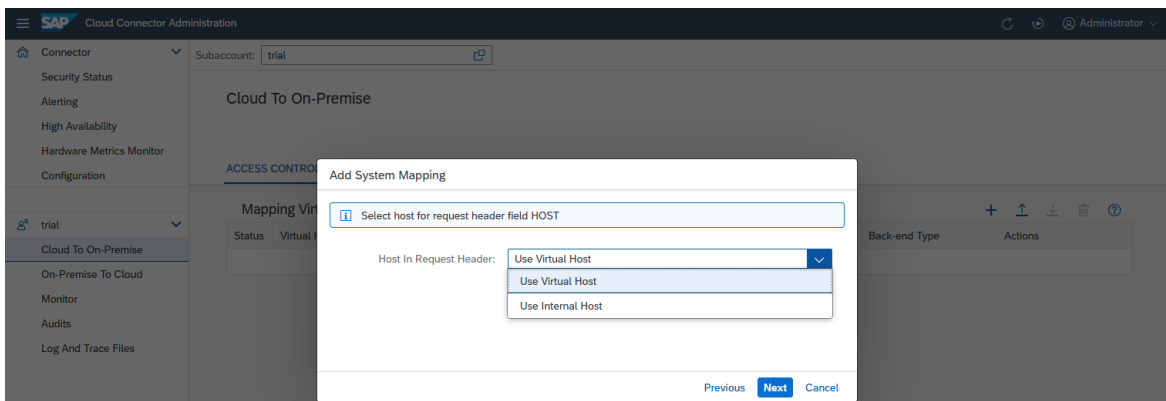
While unselected, another authentication method is used, for example, basic authentication.

Note

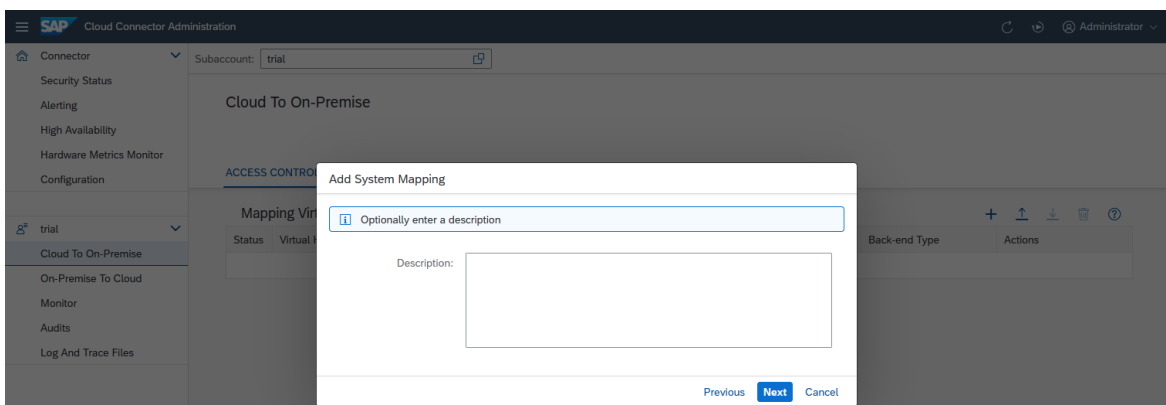
We recommend that you **keep this option deactivated**, as this lets you use principal propagation and basic authentication over the same access control entry, regardless of the logon order settings in the target system.



10. *Host In Request Header* lets you define, which host is used in the host header that is sent to the target server. By choosing **Use Internal Host**, the actual host name is used. When choosing **Use Virtual Host**, the virtual host is used. In the first case, the virtual host is still sent via the `X-Forwarded-Host` header.



11. You can enter an optional description at this stage. The respective description will be shown when pressing the *Info* button of the access control entry (table *Mapping Virtual to Internal System*).




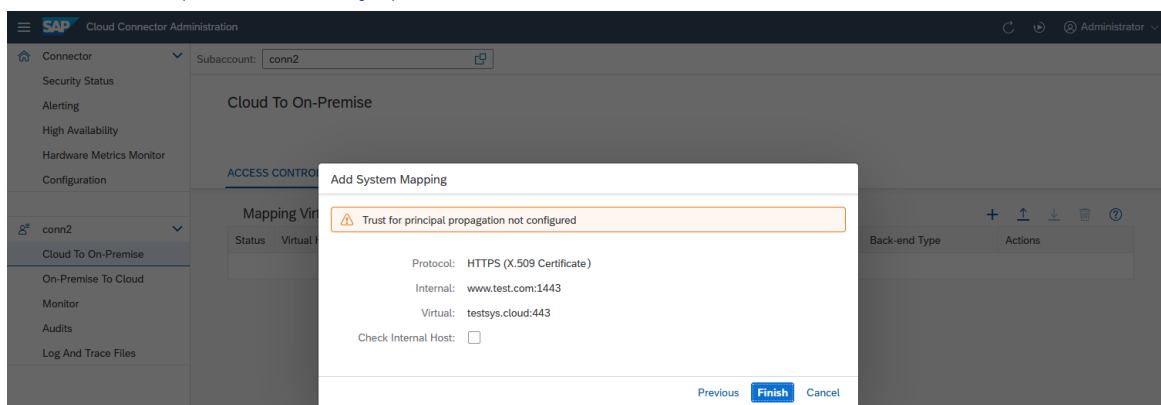
12. The summary shows information about the system to be stored and when saving the host mapping, you can trigger a ping from the Cloud Connector to the internal host, using the *Check availability of internal host* checkbox. This allows you to make sure the Cloud Connector can indeed access the internal system,

and allows you to catch basic things, such as spelling mistakes or firewall problems between the Cloud Connector and the internal host.

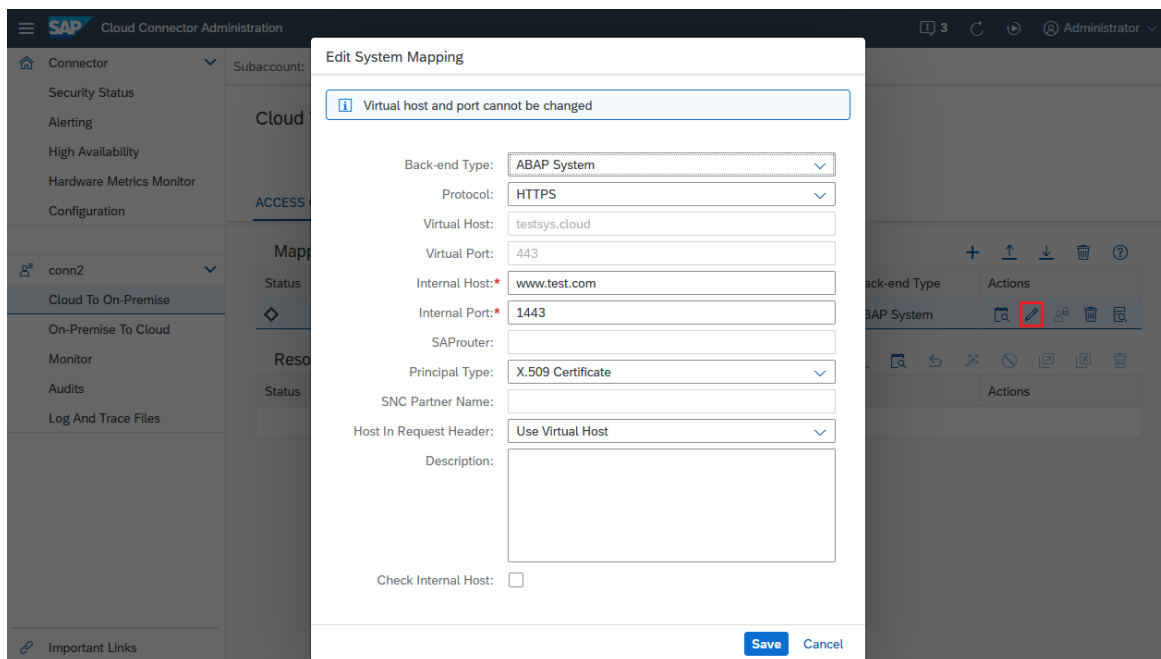
If the ping to the internal host is successful (that is, the host is reachable via TLS), the state `Reachable` is shown. If it fails, a warning pops up. You can view issue details by choosing the [Details](#) button, or check them in the log files.

This check also tries to perform client authentication if possible, regardless of the host's availability. Find additional information and hints by choosing the [Details](#) button. You can check, for example, if the system certificate acting as a client certificate is configured correctly, and if the ABAP backend trusts it.

You can execute the availability check for all selected systems in the [Access Control](#) overview by pressing the button  ([Check availability...](#)) in column *Actions*.



- Optional: You can later edit such a system mapping (via [Edit](#)) to make the Cloud Connector route the requests for `sales-system.cloud:443` to a different backend system. This can be useful if the system is currently down and there is a back-up system that can serve these requests in the meantime. However, you cannot edit the virtual name of this system mapping. If you want to use a different fictional host name in your cloud application, you must delete the mapping and create a new one.

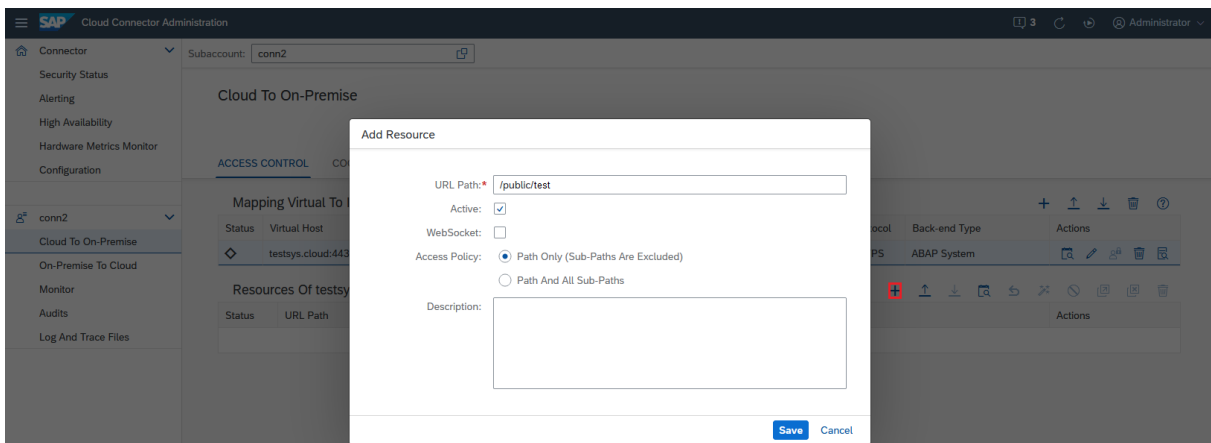


Back to [Tasks \[page 458\]](#)

Limit the Accessible Services for HTTP(S)

In addition to allowing access to a particular host and port, you also must specify which URL paths (*Resources*) are allowed to be invoked on that host. The Cloud Connector uses very strict allowlists for its access control. Only those URLs for which you explicitly granted access are allowed. All other HTTP(S) requests are denied by the Cloud Connector.

To define the permitted URLs for a particular backend system, choose the line corresponding to that backend system and choose *Add* in section *Resources Accessible On...* below. A dialog appears prompting you to enter the specific URL path that you want to allow to be invoked.



The Cloud Connector checks that the path part of the URL (up to but not including a possible question mark (?) that may denote the start of optional CGI-style query parameters) is exactly as specified in the configuration. If it is not, the request is denied. If you select option *Path and all sub-paths*, the Cloud Connector allows all requests for which the URL path (not considering any query parameters) starts with the specified string.

The *Active* checkbox lets you specify, if that resource is initially enabled or disabled. See the section below for more information on enabled and disabled resources.






The *WebSocket Upgrade* checkbox lets you specify, whether that resource allows a protocol upgrade.

Back to [Tasks \[page 458\]](#)

Activate or Suspend Resources

In some cases, it is useful for testing purposes to temporarily disable certain resources without having to delete them from the configuration. This allows you to easily reprovide access to these resources at a later point of time without having to type in everything once again.

- To suspend a resource, select it and choose the *Suspend* button:
The status icon turns red, and from now on, the Cloud Connector will deny all requests coming in for this resource.

Status	URL Path	Access Policy	Actions
<input type="checkbox"/>	/public/test	Path Only (Sub-Paths Are Excluded)	    

- To activate the resource again, select it and choose the *Activate* button.
- By choosing *Allow WebSocket upgrade/Disallow WebSocket upgrade* this is possible for the protocol upgrade setting as well.
- It is also possible to mark multiple lines and then suspend or activate all of them in one go by clicking the *Activate/Suspend* icons in the top row. The same is true for the corresponding *Allow WebSocket upgrade/Disallow WebSocket* icons.

Examples:

- */production/accounting* and *Path only (sub-paths are excluded)* are selected. Only requests of the form `GET /production/accounting` or `GET /production/accounting?name1=value1&name2=value2...` are allowed. (GET can also be replaced by POST, PUT, DELETE, and so on.)
- */production/accounting* and *Path and all sub-paths* are selected. All requests of the form `GET /production/accounting-plus-some-more-stuff-here?name1=value1...` are allowed.
- */* and *Path and all sub-paths* are selected. All requests to this server are allowed.

Back to [Tasks \[page 458\]](#)

Related Information

[Configure Domain Mappings for Cookies \[page 618\]](#)

[Consume Backend Systems \(Java Web or Java EE 6 Web Profile\)](#)

1.2.2.4.2 Configure Access Control (RFC)

Specify the backend systems that can be accessed by your cloud applications using RFC.

Tasks

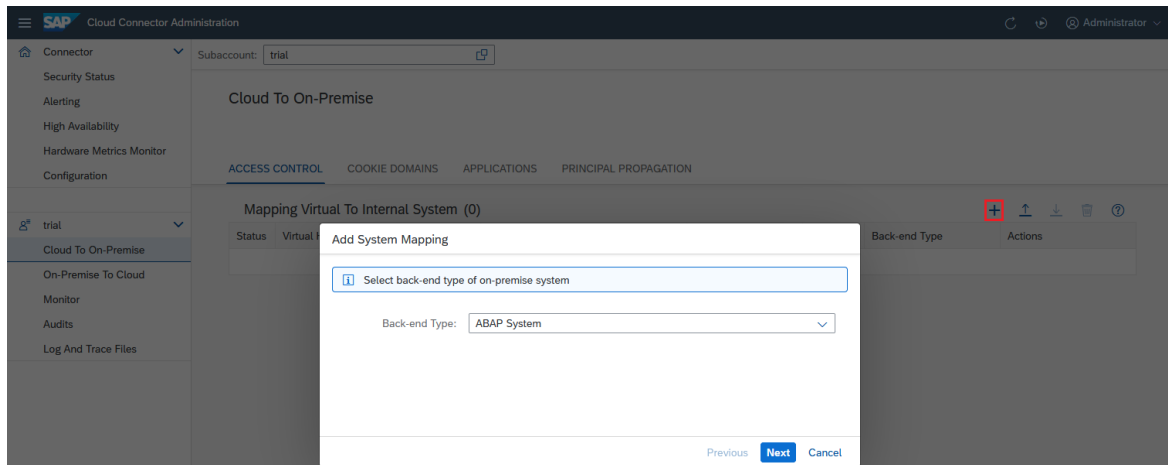
[Expose Intranet Systems \[page 466\]](#)

[Limit the Accessible Resources for RFC \[page 471\]](#)

Expose Intranet Systems

To allow your cloud applications to access a certain backend system on the intranet, insert a new entry in the Cloud Connector *Access Control* management.

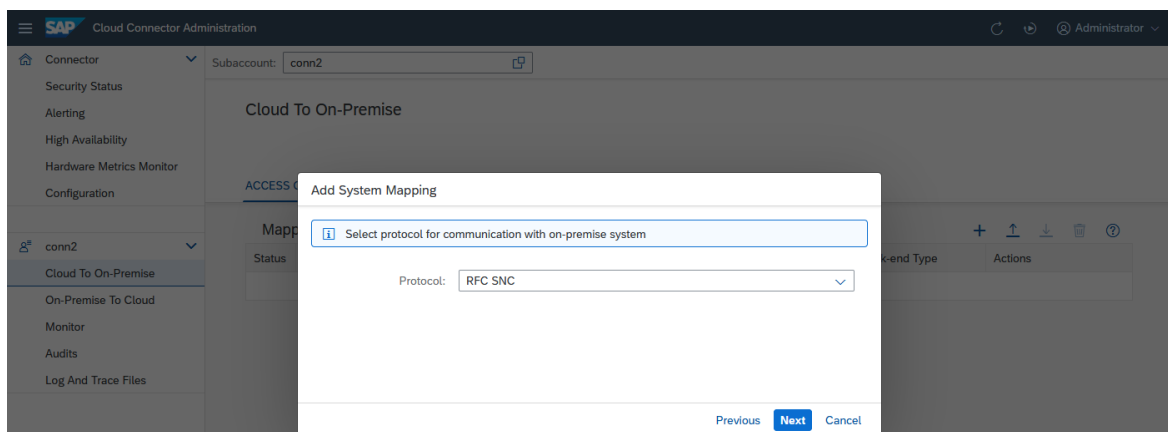
1. Choose *Cloud To On-Premise* from your *Subaccount* menu and go to tab *Access Control*.
2. Choose *Add*.
3. *Backend Type*: Select the backend system type (**ABAP System** or **SAP Gateway** for RFC).



4. Choose *Next*.
5. *Protocol*: Choose **RFC** or **RFC SNC** for connecting to the backend system.

Note

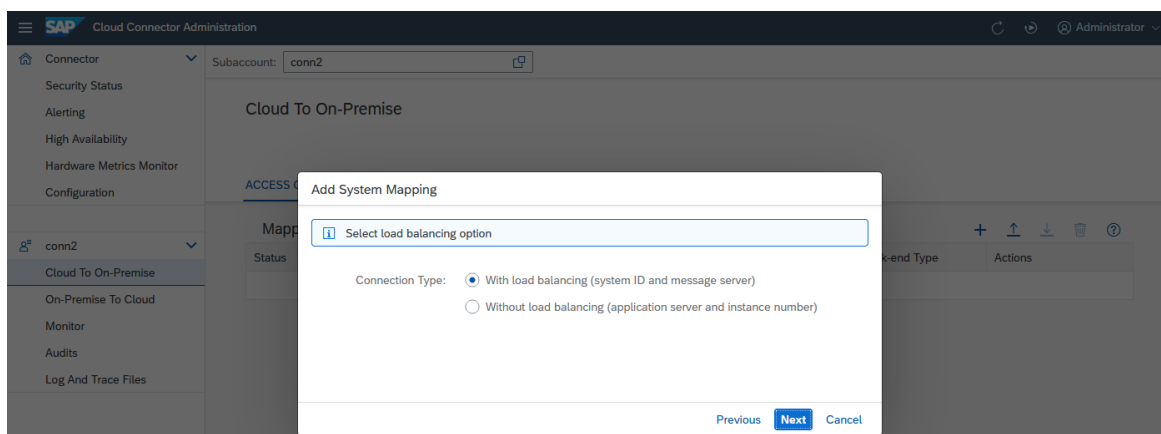
The value **RFC SNC** is independent from your settings on the cloud side, since it only specifies the communication between Cloud Connector and backend system. Using **RFC SNC**, you can ensure that the entire connection from the cloud application to the actual backend system (provided by the SSL tunnel) is secured, partly with SSL and partly with SNC. For more information, see [Initial Configuration \(RFC\) \[page 398\]](#).



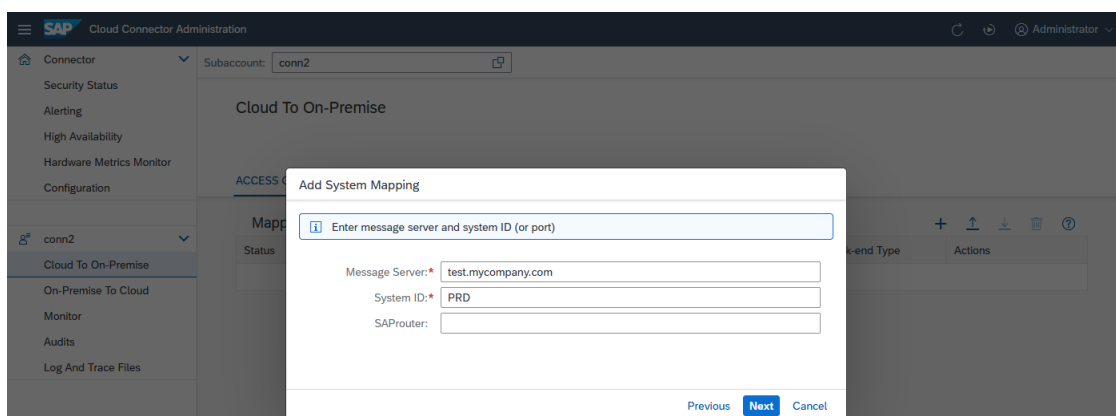
Note

- The back end must be properly configured to support SNC connections.
- SNC configuration must be provided in the Cloud Connector.

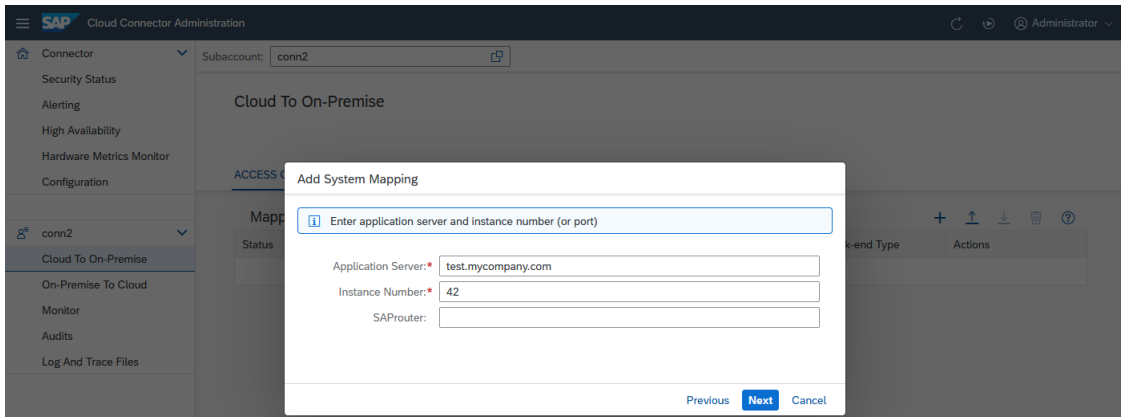
6. Choose *Next*.
7. Choose whether you want to configure a load balancing logon or connect to a specific application server.



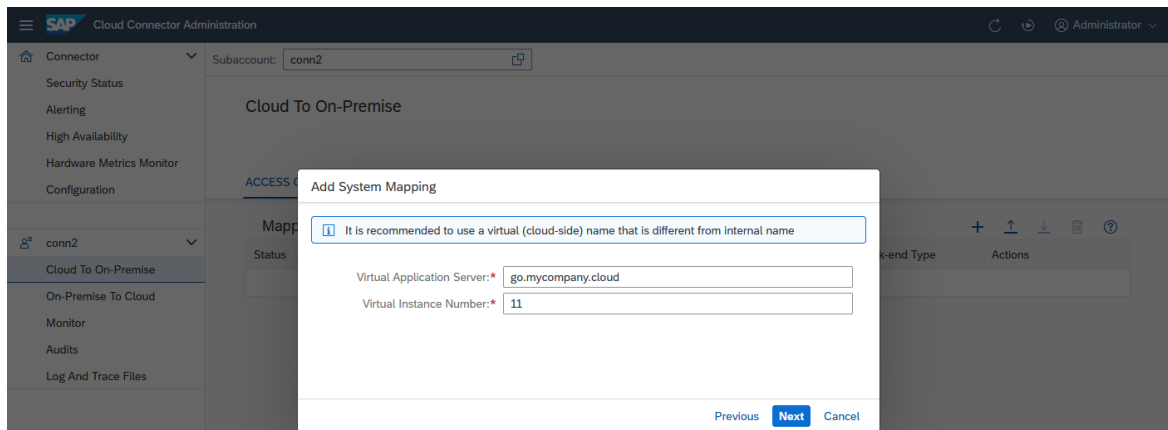
8. Specify the parameters of the backend system. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. If this is only possible using a valid *SAProuter*, specify the router in the respective field. The Cloud Connector will try to establish a connection to this system, so the address has to be real.
 - When using a load-balancing configuration, the *Message Server* specifies the message server of the ABAP system. The system ID is a three-char identifier that is also found in the SAP Logon configuration. Alternatively, it's possible to directly specify the message server port in the *System ID* field.



- When using direct logon, the *Application Server* specifies one application server of the ABAP system. The instance number is a two-digit number that is also found in the SAP Logon configuration. Alternatively, it's possible to directly specify the gateway port in the *Instance Number* field.



9. Optional: You can virtualize the system information in case you like to hide your internal host names from the cloud. The virtual information can be a fake name which does not need to exist. The fields will be pre-populated with the values of the configuration provided in *Message Server* and *System ID*, or *Application Server* and *Instance Number*.

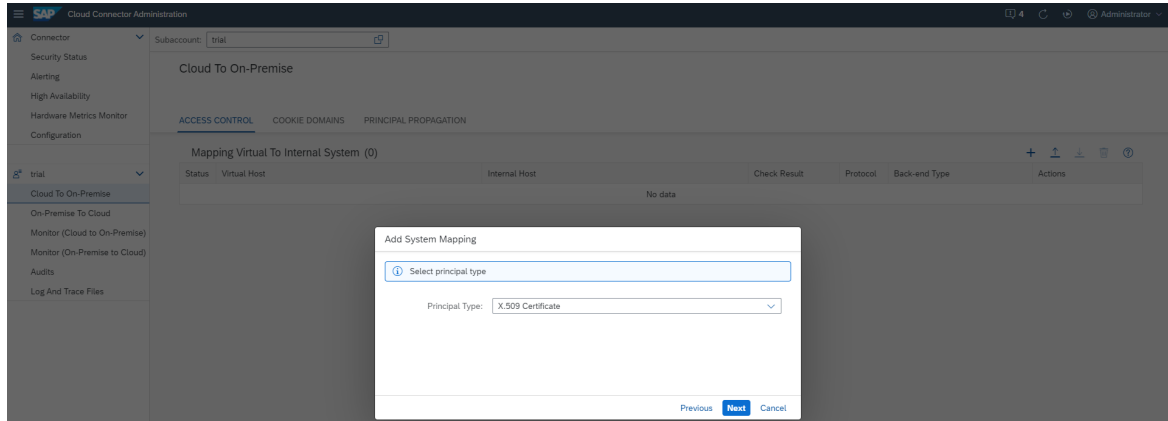


- *Virtual Message Server* - specifies the host name exactly as specified as the `jco.client.mshost` property in the RFC destination configuration in the cloud. The *Virtual System ID* allows you to distinguish between different entry points of your backend system that have different sets of access control settings. The value needs to be the same like for the `jco.client.r3name` property in the RFC destination configuration in the cloud.
 - *Virtual Application Server* - it specifies the host name exactly as specified as the `jco.client.ashost` property in the RFC destination configuration in the cloud. The *Virtual Instance Number* allows you to distinguish between different entry points of your backend system that have different sets of access control settings. The value needs to be the same like for the `jco.client.sysnr` property in the RFC destination configuration in the cloud.
10. This step is only relevant if you have chosen **RFC SNC**. The `<Principal Type>` field defines what kind of principal is used when configuring a destination on the cloud side using this system mapping with authentication type `Principal Propagation`. No matter what you choose, make sure that the general configuration for the `<Principal Type>` has been done to make it work correctly. For destinations using different authentication types, this setting is ignored. In case you choose **None** as `<Principal Type>`, it is not possible to apply principal propagation to this system.

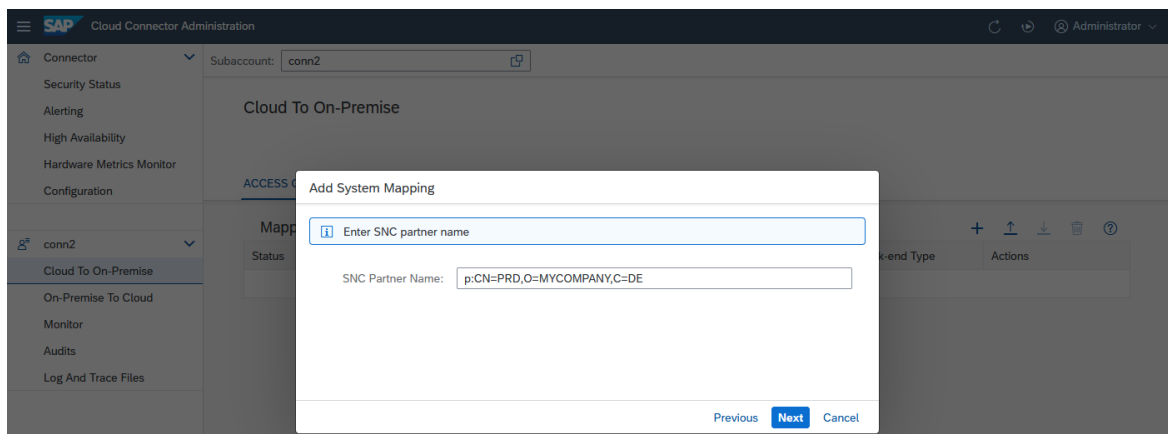
Note

If you use an RFC connection, you cannot choose between different principal types. Only the **x.509** certificate is supported. You need an SNC-enabled backend connection to use it.

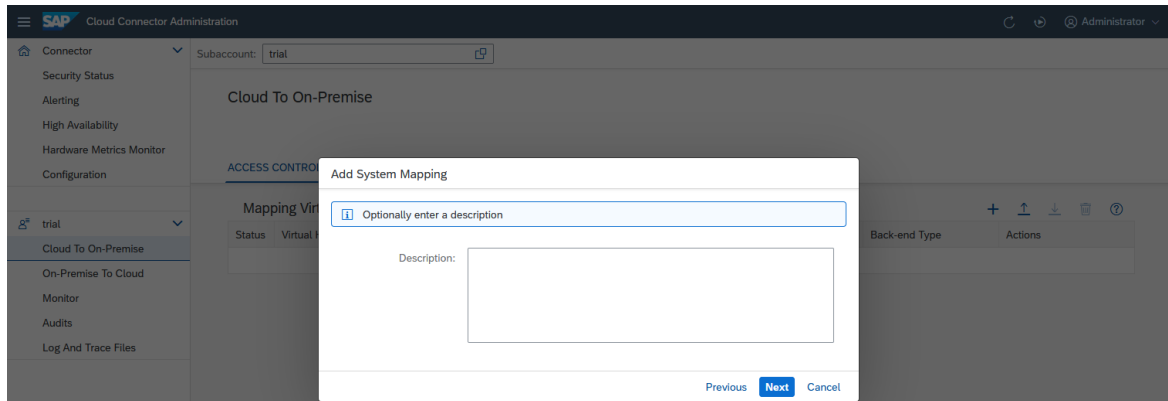
For more information on principal propagation, see [Configuring Principal Propagation \[page 420\]](#).



11. **SNC Partner Name:** This step will only come up if you have chosen **RFC SNC**. The SNC partner name needs to contain the correct SNC identification of the target system.



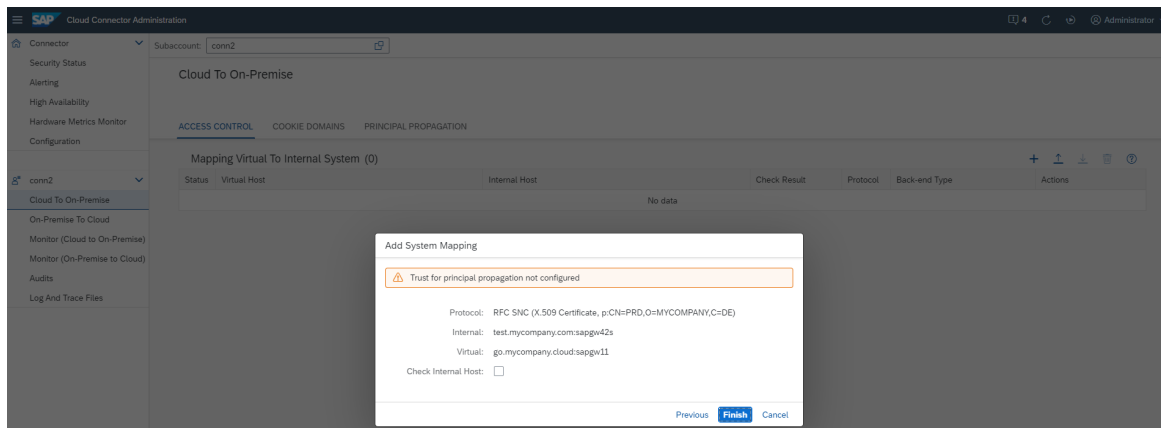
12. **Mapping Virtual to Internal System:** You can enter an optional description at this stage. The respective description will be shown as a rich tooltip when the mouse hovers over the entries of the virtual host column (table).



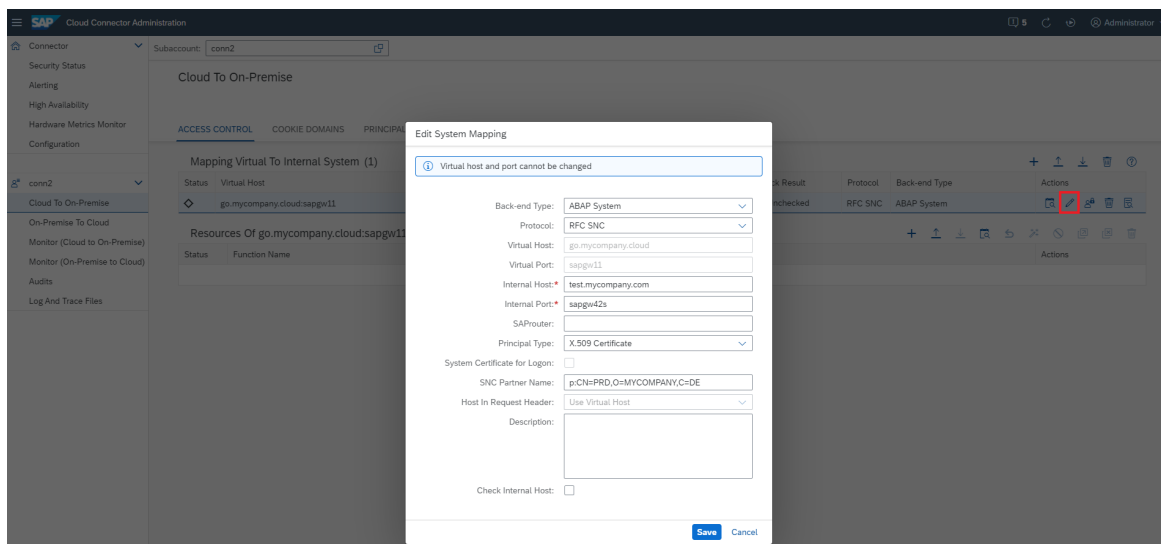
13. The summary shows information about the system to be stored. When saving the system mapping, you can trigger a ping from the Cloud Connector to the internal host, using the *Check Internal Host* checkbox. This allows you to make sure the Cloud Connector can indeed access the internal system, and allows you to catch basic things, such as spelling mistakes or firewall problems between the Cloud Connector and the internal host.

If the ping to the internal host is successful (that is, the host is reachable via TLS), the state *Reachable* is shown. If it fails, a warning pops up. You can view issue details by choosing the *Details* button, or check them in the log files.

You can execute such a check at any time later for all selected systems in the *Access Control* overview.



14. Optional: You can later edit a system mapping (choose *Edit*) to make the Cloud Connector route the requests for **sales-system.cloud:sapgw42** to a different backend system. This can be useful if the system is currently down and there is a back-up system that can serve these requests in the meantime. However, you cannot edit the virtual name of this system mapping. If you want to use a different fictional host name in your cloud application, you must delete the mapping and create a new one. Here, you can also change the *Principal Type* to **None** in case you don't want to allow principal propagation to a certain system.

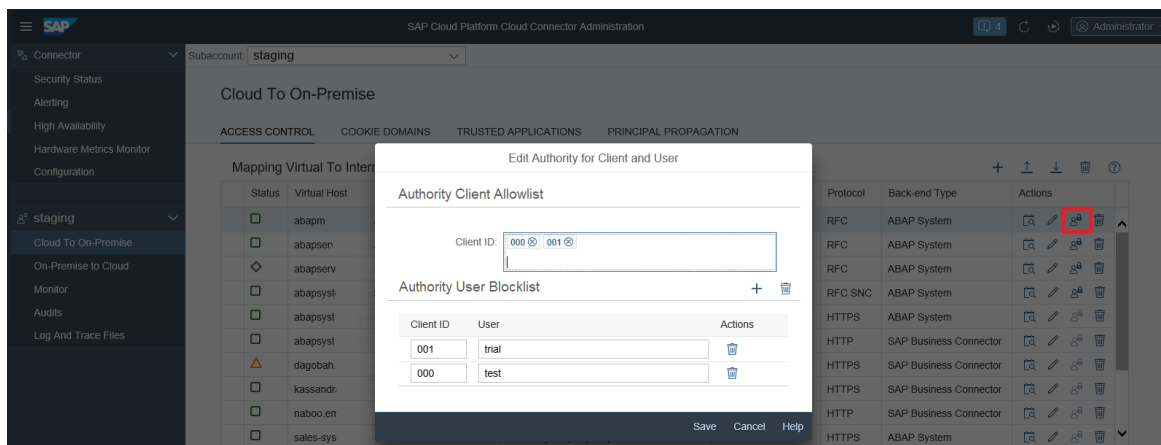


15. Optional. You can later edit a system mapping to add more protection to your system when using RFC via the Cloud Connector, by restricting the mapping to specified clients and users: in column *Actions*, choose the button *Maintain Authority Lists (only RFC)* to open an allowlist/blocklist dialog. In section *Authority Client Allowlist*, enter all clients of the corresponding system in the field **<Client ID>** that you want to allow to use the Cloud Connector connection. In section *Authority User Blocklist*, press the button *Add a*

user authority (+) to enter all users you want to exclude from this connection. Each user must be assigned to a specified client. When you are done, press *Save*.

Note

This function applies for RFC/RFC SNC only.

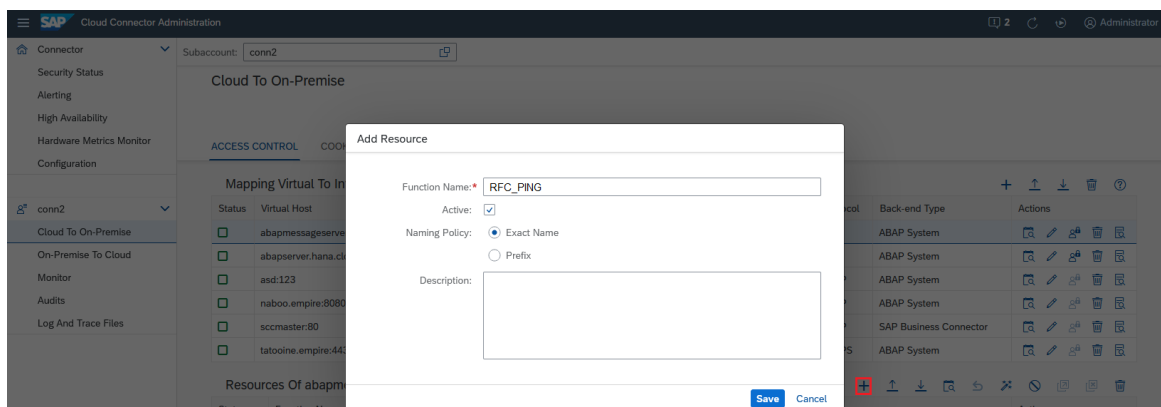


Back to [Tasks \[page 465\]](#)

Limit the Accessible Resources for RFC

In addition to allowing access to a particular host and port, you also must specify which function modules (*Resources*) are allowed to be invoked on that host. You can enter an optional description at this stage. The Cloud Connector uses very strict allowlists for its access control. Besides internally used infrastructure function modules, only function modules for which you explicitly granted access are allowed.

1. To define the permitted function modules for a particular backend system, choose the row corresponding to that backend system and press *Add* in section *Resources Accessible On...* below. A dialog appears, prompting you to enter the specific function module name whose invoking you want to allow.



2. The Cloud Connector checks that the function module name of an incoming request is exactly as specified in the configuration. If it is not, the request is denied.
3. If you select the *Prefix* option, the Cloud Connector allows all incoming requests, for which the function module name begins with the specified string.

- The *Active* checkbox allows you to specify whether that resource should be initially enabled or disabled.

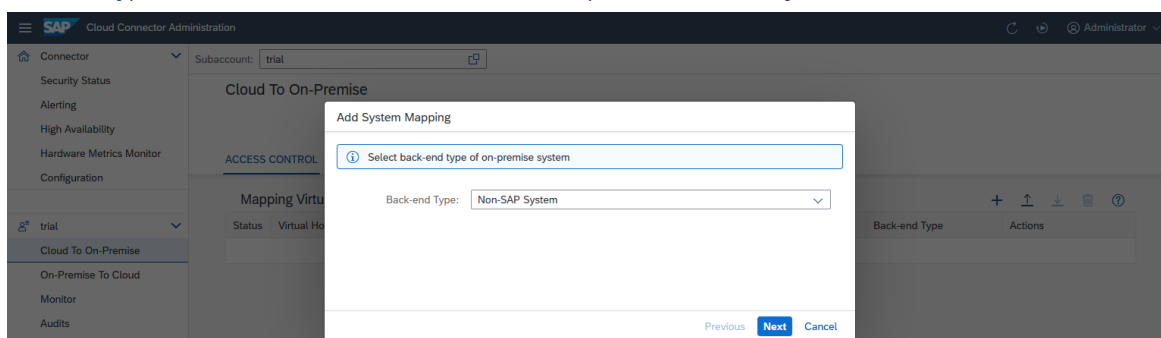
Back to [Tasks \[page 465\]](#)

1.2.2.4.3 Configure Access Control (LDAP)

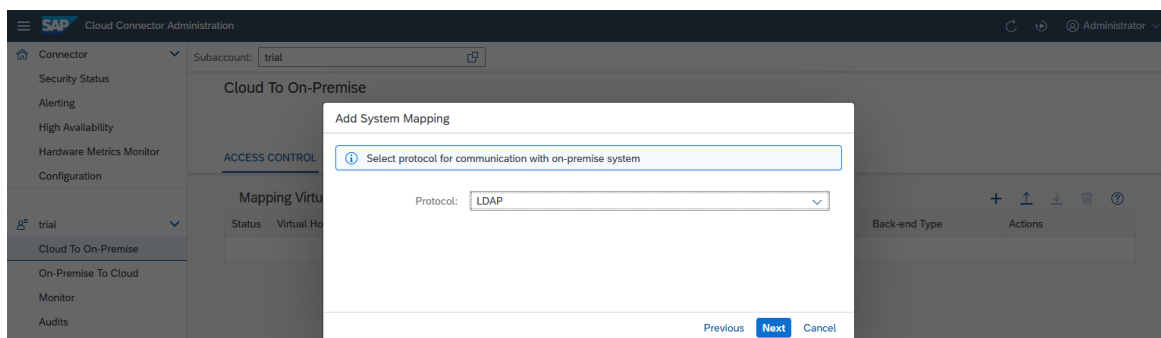
Add a specified system mapping to the Cloud Connector if you want to use an on-premise LDAP server or user authentication in your cloud application.

To allow your cloud applications to access an on-premise LDAP server, insert a new entry in the Cloud Connector access control management.

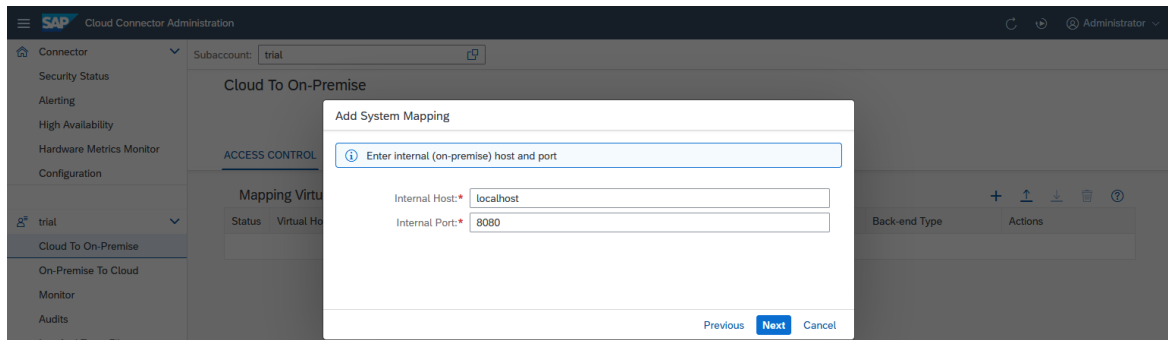
- Choose *Cloud To On-Premise* from your *Subaccount* menu.
- Choose *Add (+)*. A wizard opens and asks for the required values.
- Backend Type*: Select **Non-SAP System** from the drop down list. When you are done, choose *Next*.



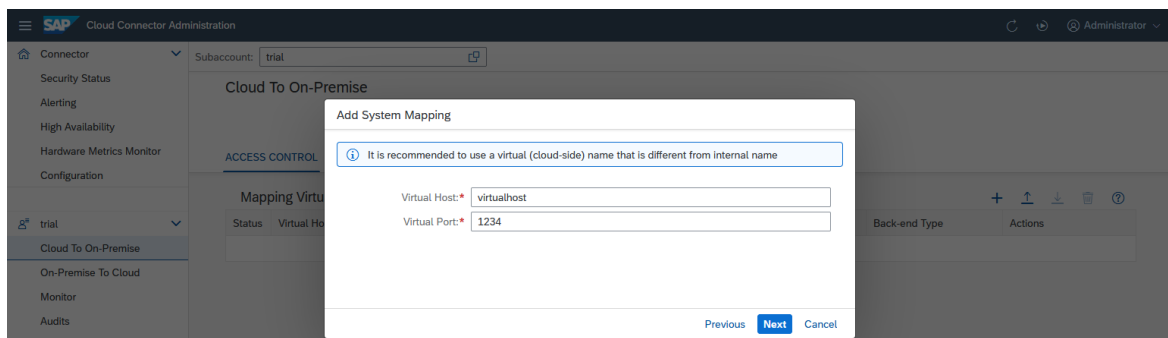
- Protocol*: Select **LDAP** or **LDAPS** for the connection to the backend system. When you are done, choose *Next*.



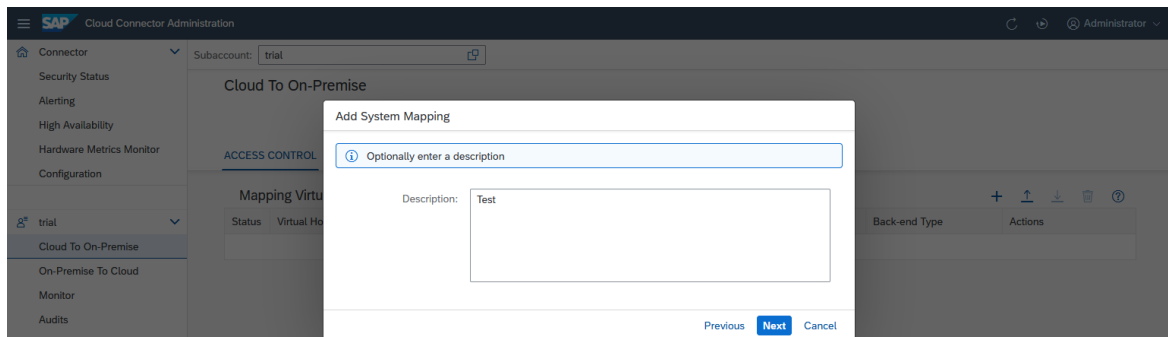
- Internal Host* and *Internal Port*: specify the host and port under which the target system can be reached within the intranet. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. The Cloud Connector will try to forward the request to the network address specified by the internal host and port, so this address needs to be real.



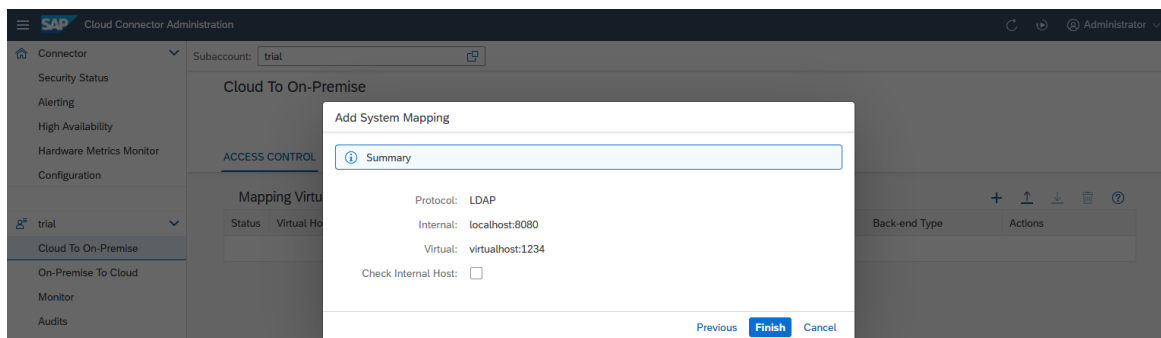
6. Enter a *Virtual Host* and *Virtual Port*. The virtual host can be a fake name and does not need to exist. The fields are pre-populated with the values of the *Internal Host* and *Internal Port*.



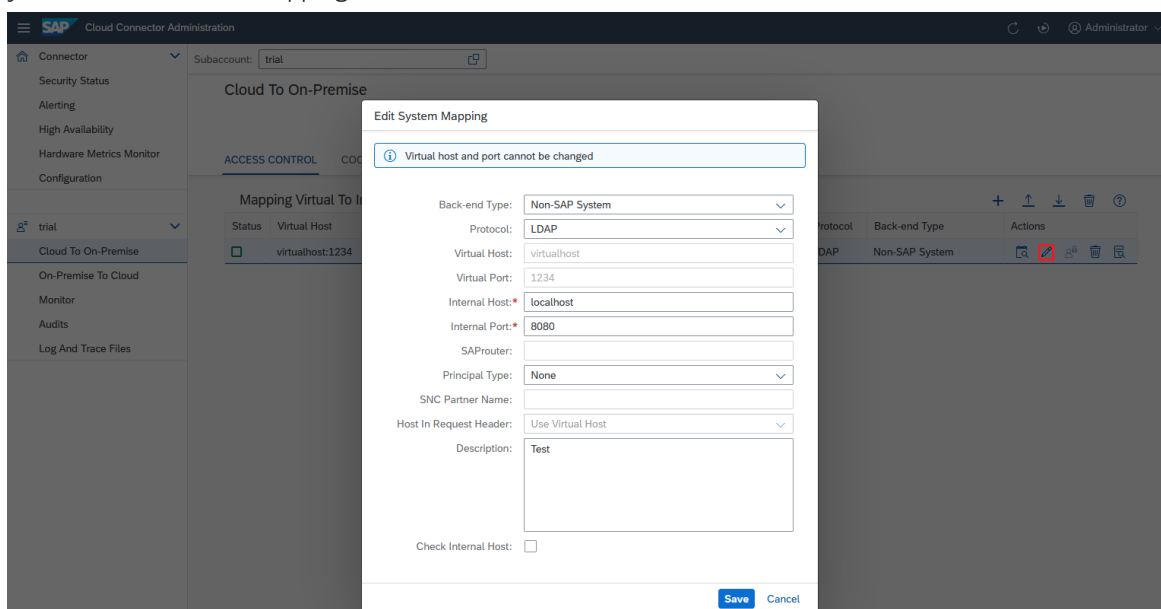
7. You can enter an optional description at this stage. The respective description will be shown as a tooltip when you press the button *Show Details* in column *Actions* of the *Mapping Virtual To Internal System* overview.



8. The summary shows information about the system to be stored. When saving the host mapping, you can trigger a ping from the Cloud Connector to the internal host, using the *Check Internal Host* check box. This allows you to make sure the Cloud Connector can indeed access the internal system. Also, you can catch basic things, such as spelling mistakes or firewall problems between Cloud Connector the internal host. If the ping to the internal host is successful, the state `reachable` is shown. If it fails, a warning is displayed in column **Check Result**. You can view issue details by choosing the *Details* button, or check them in the log files.
You can execute such a check at any time later for all selected systems in the *Mapping Virtual To Internal System* overview by pressing *Check Availability of Internal Host* in column *Actions*.



- Optional: You can later edit the system mapping (by choosing *Edit*) to make the Cloud Connector route the requests to a different LDAP server. This can be useful if the system is currently down and there is a back-up LDAP server that can serve these requests in the meantime. However, you cannot edit the virtual name of this system mapping. If you want to use a different fictional host name in your cloud application, you have to delete the mapping and create a new one.

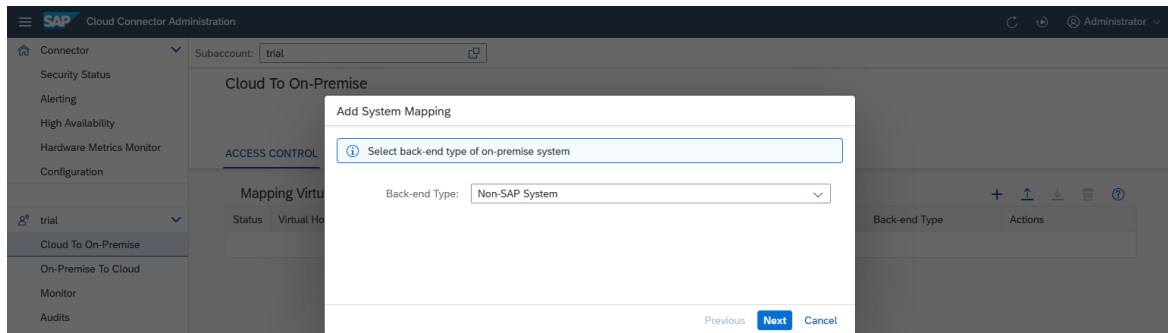


1.2.2.4.4 Configure Access Control (TCP)

Add a specified system mapping to the Cloud Connector if you want to use the TCP protocol for communication with a backend system.

To allow your cloud applications to access a certain backend system on the intranet via TCP, insert a new entry in the Cloud Connector access control management.

- Choose *Cloud To On-Premise* from your *Subaccount* menu.
- Choose *Add* (+). A wizard opens and asks for the required values.
- Backend Type*: Select an appropriate system type, for example, **Non-SAP System**, from the drop-down list. When you are done, choose *Next*.

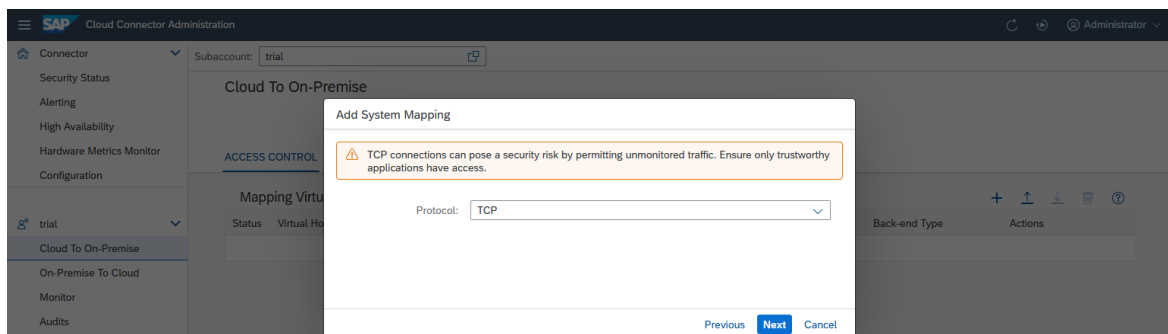


4. **Protocol:** Select **TCP** or **TCP SSL** for the connection to the backend system. When choosing TCP, you can perform an end-to-end TLS handshake from the cloud client to the backend. If the cloud-side client is using plain communication, but you still need to encrypt the hop between Cloud Connector and the backend, choose **TCP SSL**. When you are done, choose **Next**.

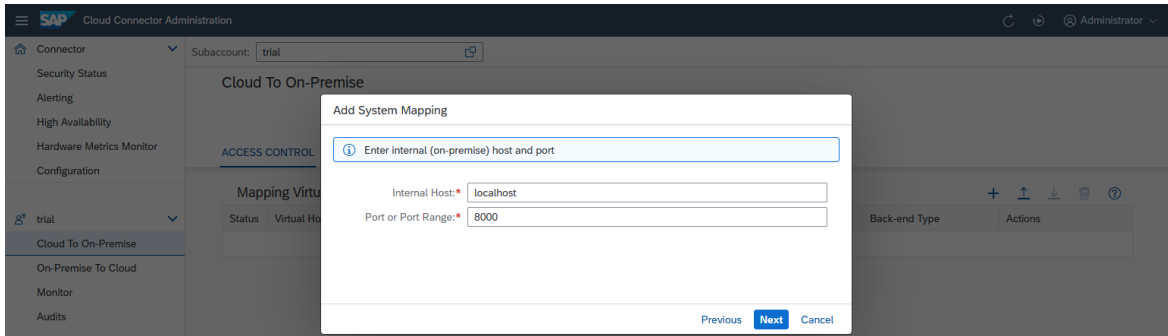
Note

When selecting TCP as protocol, the following warning message is displayed: TCP connections can pose a security risk by permitting unmonitored traffic. Ensure only trustworthy applications have access. The reason is that using plain TCP, the Cloud Connector cannot see or log any detail information about the calls. Therefore, in contrast to HTTP or RFC (both running on top of TCP), the Cloud Connector cannot check the validity of a request. To minimize this risk, make sure you

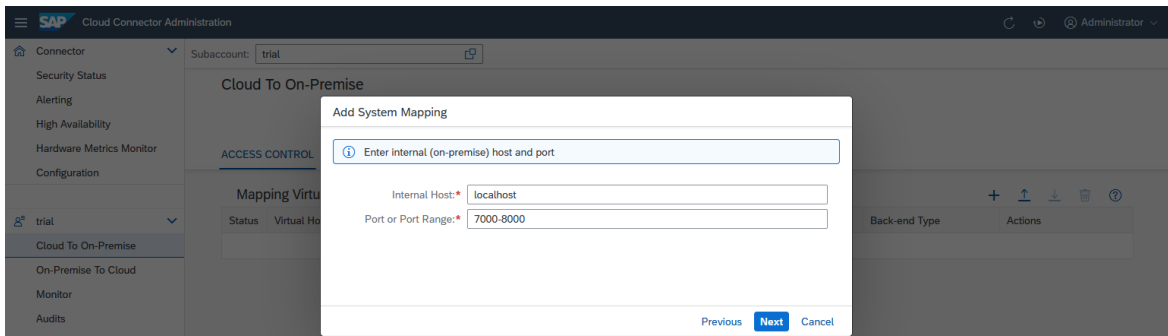
- deploy only trusted applications on SAP BTP.
- configure an application allowlist in the Cloud Connector, see [Set Up Trust \[page 420\]](#).
- take the recommended security measures for your SAP BTP (sub)account. See section [Security](#) in the SAP BTP documentation.



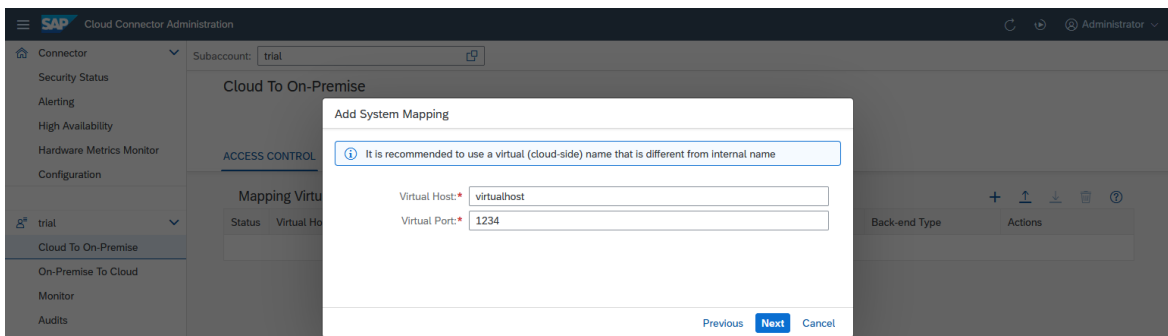
5. **Internal Host** and **Port or Port Range:** specify the host and port under which the target system can be reached within the intranet. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. The Cloud Connector will try to forward the request to the network address specified by the internal host and port. That is why this address needs to be real.



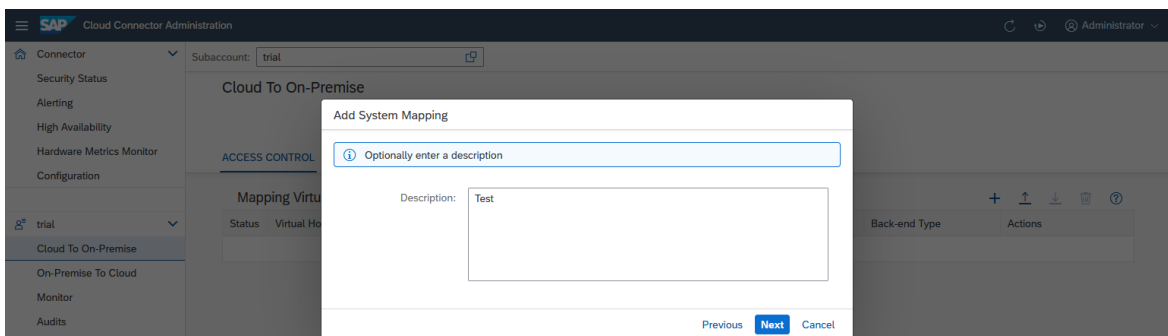
For TCP and TCP SSL, you can also specify a port range through its lower and upper limit, separated by a hyphen.



6. Enter a *Virtual Host* and *Virtual Port*. The virtual host can be a fake name and does not need to exist. The fields are prepopulated with the values of the *Internal Host* and *Port or Port Range*.



7. You can enter an optional description at this stage. The respective description will be shown as a tooltip when you press the button *Show Details* in column *Actions* of the *Mapping Virtual To Internal System* overview.

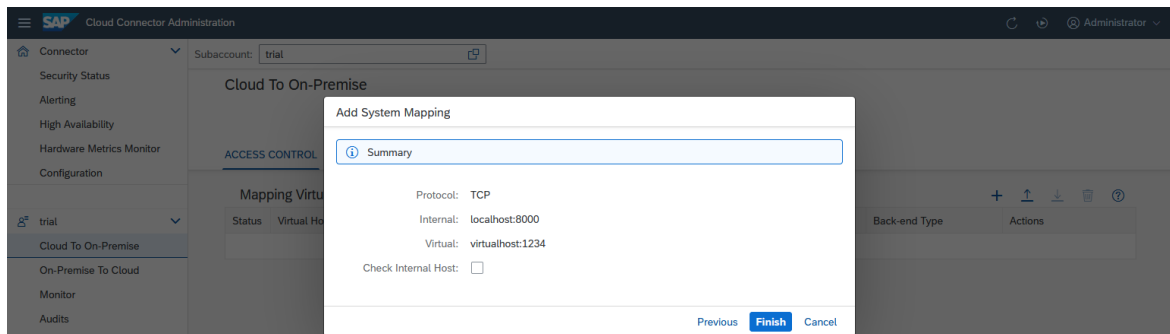


8. The summary shows information about the system to be stored. When saving the host mapping, you can trigger a ping from the Cloud Connector to the internal host, using the *Check Internal Host* checkbox. This

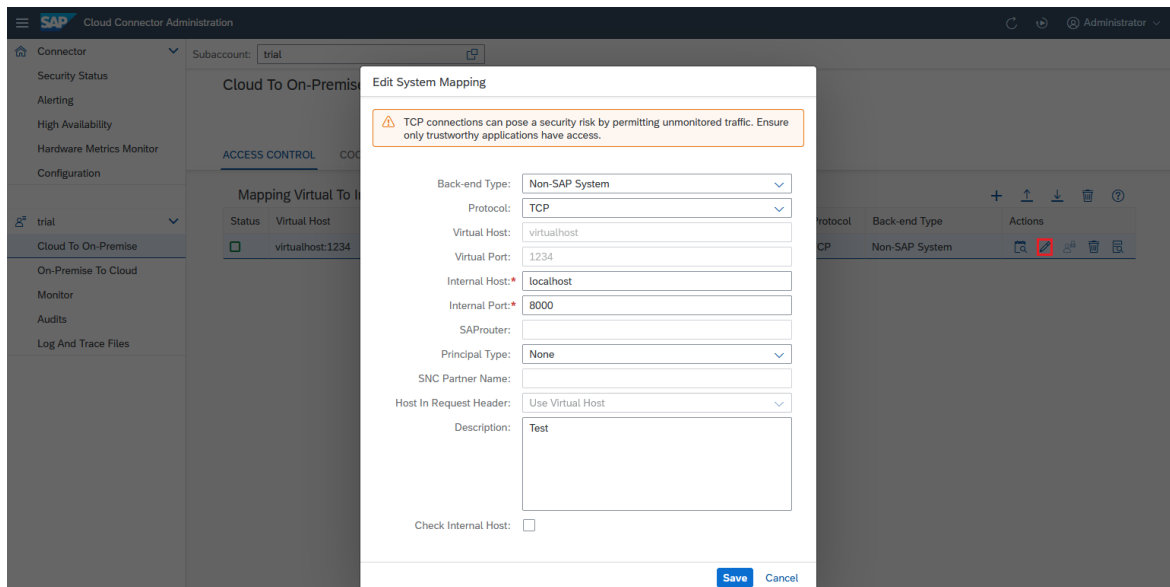
allows you to make sure the Cloud Connector can access the internal system. Also, you can catch basic things, such as spelling mistakes or firewall problems between the Cloud Connector and the internal host. If the ping to the internal host is successful (that is, the host is reachable via TLS), the state `Reachable` is shown. If it fails, a warning is displayed in column **Check Result**. You can view issue details by choosing the [Details](#) button, or check them in the log files.

This check also tries to perform client authentication, if possible for TCPS, regardless of the host's availability. Find additional information and hints by choosing the [Details](#) button. You can check, for example, if the system certificate acting as a client certificate is configured correctly, and if the backend trusts it.

You can execute such a check at any time later for all selected systems in the [Mapping Virtual To Internal System](#) overview by pressing [Check Availability of Internal Host](#) in column *Actions*.



9. Optional: You can later edit the system mapping (by choosing [Edit](#)) to make the Cloud Connector route the requests to a different backend system. This can be useful if the system is currently down and there is a backup system that can serve these requests in the meantime. However, you cannot edit the virtual name nor port of this system mapping. If you want to use a different fictional host name in your cloud application, you must delete the mapping and create a new one. The same goes for port ranges. If a port range needs to be changed, you must delete the mapping and create it again with the desired port range.



1.2.2.4.5 Configure Accessible Resources

Configure backend systems and resources in the Cloud Connector, to make them available for a cloud application.

Tasks

[Map Systems and Limit Access \[page 478\]](#)

[Use Scenarios for Resources \[page 480\]](#)

Map Systems and Limit Access

Initially, after installing a new Cloud Connector, no network systems or resources are exposed to the cloud. You must configure each system and resource used by applications of the connected cloud subaccount. To do this, choose *Cloud To On Premise* from your subaccount menu and go to tab *Access Control*:

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar contains navigation options like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. The main area is titled 'Cloud To On-Premise' and has tabs for 'ACCESS CONTROL', 'COOKIE DOMAINS', 'APPLICATIONS', and 'PRINCIPAL PROPAGATION'. The 'ACCESS CONTROL' tab is active, showing a table of 'Mapping Virtual To Internal System (7)'. Below this, a red box highlights a table titled 'Resources Of abapmessageserver.hana.cloud:sapmsFCB (4)'. The table lists resources with their status, function names, and naming policies.

Status	Virtual Host	Internal Host	Check Result	Protocol	Back-end Type	Actions
<input type="checkbox"/>	abapmessageserver.hana.cloud:sapmsFCB	xxxxx.wdf.sap.corp:sapmsV9U	◇ Unchecked	RFC	ABAP System	
<input type="checkbox"/>	abapserver.hana.cloud:sapgw42	xxxxx.wdf.sap.corp:sapgw51	◇ Unchecked	RFC	ABAP System	
<input type="checkbox"/>	asd:123	asd:123	◇ Unchecked	HTTP	ABAP System	
<input type="checkbox"/>	ldciv9u.wdf.sap.corp:sapgw51	xxxxx.wdf.sap.corp:sapgw51	◇ Unchecked	RFC	ABAP System	
<input type="checkbox"/>	naboo.empire:8080	xxxxx.dhcp.wdf.sap.corp:5555	◇ Unchecked	HTTP	ABAP System	
<input type="checkbox"/>	sccmaster:80	xxxxx.wdf.sap.corp:443	◇ Unchecked	HTTP	SAP Business Connector	
<input type="checkbox"/>	tatooline.empire:443	xxxxx.dhcp.wdf.sap.corp:4443	◇ Unchecked	HTTP	ABAP System	

Status	Function Name	Naming Policy	Actions
<input type="checkbox"/>	JTEST_ADD_INT	Exact Name	
<input type="checkbox"/>	RFC_RAISE	Prefix	
<input type="checkbox"/>	SBC_STRING	Exact Name	
<input type="checkbox"/>	SFTC_CONNECTION	Exact Name	

The Cloud Connector supports any type of system (SAP or non-SAP system) that can be called via one of the supported protocols. For example, a convenient way to access an ABAP system from a cloud application is via [SAP NetWeaver Gateway](#), as it allows the consumption of ABAP content via HTTP and open standards.

- For systems using HTTP communication, see: [Configure Access Control \(HTTP\) \[page 457\]](#).
- For information on configuring RFC resources, see: [Configure Access Control \(RFC\) \[page 465\]](#).

We recommend that you limit the access to backend services and resources. Instead of configuring a system and granting access to all its resources, grant access only to the resources needed by the cloud application. For

example, define access to an HTTP service by specifying the service URL root path and allowing access to all its subpaths.

When configuring an on-premise system, you can define a virtual host and port for the specified system. The virtual host name and port represent the fully qualified domain name of the related system in the cloud. We recommend that you use the virtual host name/port mapping to prevent leaking information about a system's physical machine name and port to the cloud.

Edit System Mapping

i Virtual host and port cannot be changed

Back-end Type:	ABAP System
Protocol:	RFC
Virtual Host:	abapmessageserver.hana.cloud
Virtual Port:	sapmsFCB
Internal Host:*	xxxxx.wdf.sap.corp
Internal Port:*	sapmsV9U
SAProuter:	
Principal Type:	None
SNC Partner Name:	
Host In Request Header:	Use Virtual Host
Description:	
Check Internal Host:	<input type="checkbox"/>

Save

Cancel

[Back to Tasks \[page 478\]](#)

Use Scenarios for Resources

The Cloud Connector lets you define a set of resources as a scenario that you can export, and import into another Cloud Connector.

Scenarios are useful if you provide an application to many consumers, which invokes a large number of resources in an on-premise system. In this case, you must expose a system on your Cloud Connector that covers all required resources, which increases the risk of incorrect configuration.

[Define and Export a Scenario \[page 480\]](#)

[Import a Scenario \[page 481\]](#)

[Remove a Scenario \[page 481\]](#)

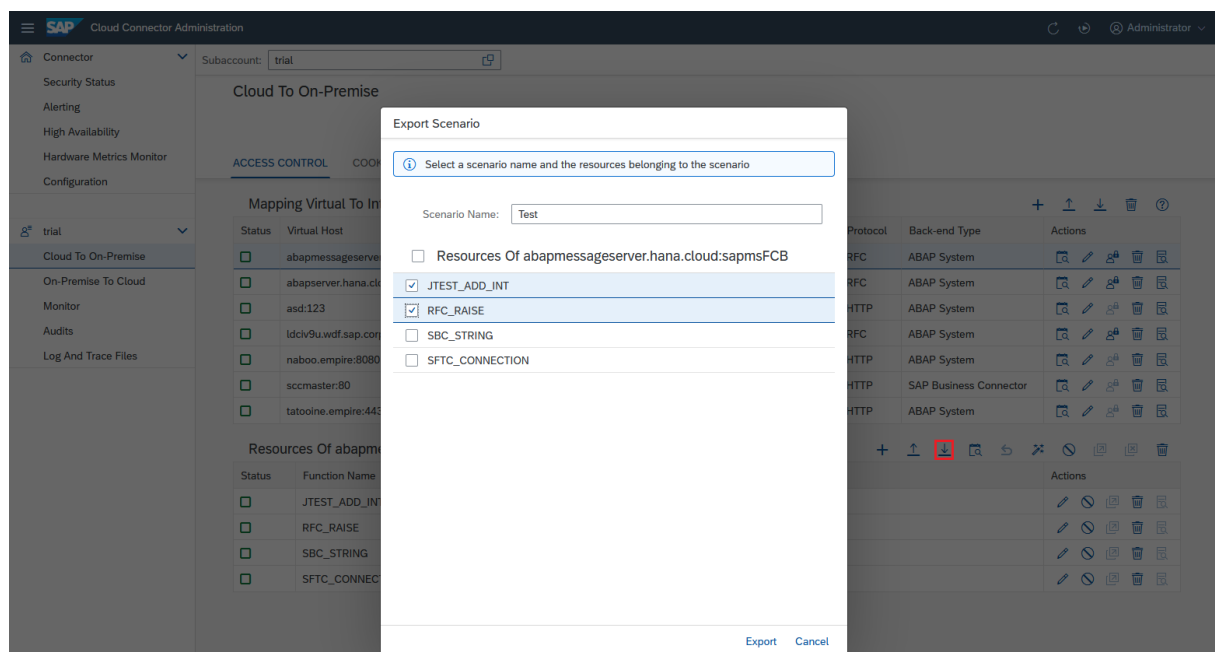
Define and Export a Scenario

If you, as application owner, have implemented and tested a scenario, and configured a Cloud Connector accordingly, you can define the scenario as follows:

1. Choose the *Export Scenario* button.
2. In the dialog, select the resources that belong to the scenario.
3. In the field `<Scenario Name>`, choose a descriptive name, under which the set of resources can be identified in the consuming Cloud Connector.
4. Choose *Export* to store the scenario.

Note

For applications provided by SAP, default scenario definitions may be available. To verify this, check the corresponding application documentation.



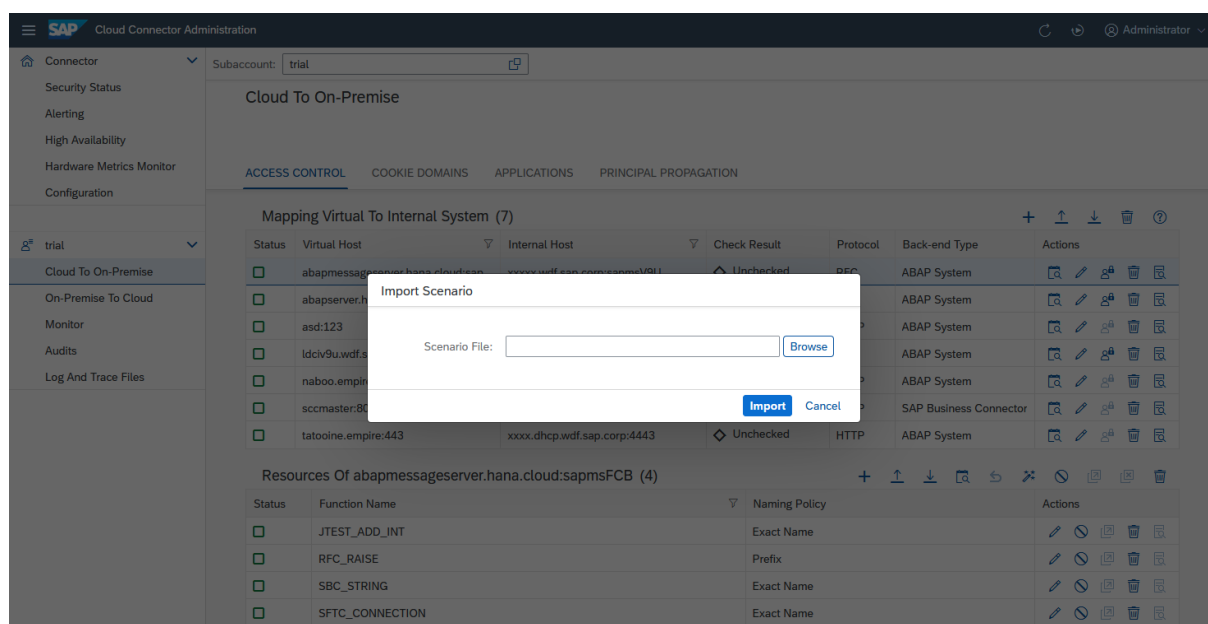
[Back to Use Scenarios for Resources \[page 480\]](#)

Back to [Tasks \[page 478\]](#)

Import a Scenario

As an administrator taking care of a scenario configuration in some other Cloud Connector, perform the following steps:

1. Choose the *Import Scenario* button to add all required resources to the desired access control entry.
2. In the dialog, navigate to the folder of the archive that contains the scenario definition.
3. Choose *Import*. The resources of the scenario are merged with the existing set of resources which are already available in the access control entry.



All resources belonging to a scenario get an additional scenario icon in their status. When hovering over it, the assigned scenario(s) of this resource are listed.

Back to [Use Scenarios for Resources \[page 480\]](#)

Back to [Tasks \[page 478\]](#)

Remove a Scenario

To remove a scenario:

1. Choose the *Remove Scenario* button.
2. In the dialog, choose the scenario(s) you want to remove.
3. Choose *Remove* to remove all resources associated with the chosen entry from the access control entry. Resources that existed before, or are assigned to another scenario as well, are not removed.

The screenshot displays the SAP Cloud Connector Administration interface. The main content area is titled 'Cloud To On-Premise' and shows a table for 'Mapping Virtual To Internal System (6)'. The table has columns for Status, Virtual Host, Internal Host, Check Result, Protocol, Back-end Type, and Actions. Below this table, there is a section for 'Resources Of asd:123 (3)' with columns for Status, URL Path, Access Policy, and Actions.

Status	Virtual Host	Internal Host	Check Result	Protocol	Back-end Type	Actions
<input type="checkbox"/>	abapmessagesserver.hana.cloud:sapms...	xxxxx.wdf.sap.corp:sapmsV9U	◇ Unchecked	RFC	ABAP System	[Icons]
<input type="checkbox"/>	abapserver.hana.cloud:sapgw42	xxxxx.wdf.sap.corp:sapgw51	◇ Unchecked	RFC	ABAP System	[Icons]
<input type="checkbox"/>	asd:123	asd:123	◇ Unchecked	HTTP	ABAP System	[Icons]
<input type="checkbox"/>	naboo.empire:8080	xxxxx.dhcp.wdf.sap.corp:5555	◇ Unchecked	HTTP	ABAP System	[Icons]
<input type="checkbox"/>	sccmaster:80	xxxx.wdf.sap.corp:443	◇ Unchecked	HTTP	SAP Business Connector	[Icons]
<input type="checkbox"/>	tatooine.empire:443	xxxx.dhcp.wdf.sap.corp:4443	◇ Unchecked	HTTP	ABAP System	[Icons]

Status	URL Path	Access Policy	Actions
<input checked="" type="checkbox"/>	/everybody	Path And All Sub-Paths	[Icons]
<input checked="" type="checkbox"/>	/public	Path Only (Sub-Paths Are Excluded)	[Icons]
<input type="checkbox"/>	/qwe	Path And All Sub-Paths	[Icons]

Back to [Use Scenarios for Resources \[page 480\]](#)

Back to [Tasks \[page 478\]](#)

1.2.2.5 Configuration REST APIs

You can use a set of APIs to perform the basic setup of the Cloud Connector.

Context

The Cloud Connector provides several REST APIs that let you configure a newly installed Cloud Connector. The configuration options correspond to the following steps:

- [Initial Configuration \[page 388\]](#)
- [Managing Subaccounts \[page 401\]](#)
- [Configure Access Control \[page 456\]](#)

Note

All configuration APIs start with the following string: `/api/v1/configuration`.

For general information on the Cloud Connector REST APIs, see also [REST APIs \[page 728\]](#).

Available APIs

- [Common Properties \[page 483\]](#)
- [High Availability Settings \[page 485\]](#)
- [Proxy Settings \[page 496\]](#)
- [Authentication and UI Settings \[page 499\]](#)
- [Certificate Management for Backend Communication \[page 511\]](#)
- [Solution Management Configuration \[page 528\]](#)
- [Backup \[page 531\]](#)
- [Subaccount \[page 533\]](#)
- [System Mappings \[page 544\]](#)
- [System Mapping Resources \[page 549\]](#)
- [Domain Mappings \[page 561\]](#)
- [Subaccount Service Channels \[page 564\]](#)
- [Access Control Entities \[page 584\]](#)

Related Information

[Examples \[page 591\]](#)

1.2.2.5.1 Common Properties

Read and edit the Cloud Connector's common properties via API.

Get Common Properties

URI	<code>/api/v1/configuration/connector</code>
Method	<code>GET</code>
Request	
Response	<code>{ha, description}</code>
Errors	

- **Response Properties:**

- `ha`: Cloud Connector instance assigned to a high-availability role. Its value is either the string *master* or *shadow*.
- `description`: Description of the Cloud Connector.

Get Version

Note

Available as of version 2.14.0.

URI	<code>/api/v1/connector/version</code>
Method	<code>GET</code>
Request	
Response	<code>{version}</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

- **Response Properties:**

`version`: A string; the Cloud Connector version.

Set Description (Master Only)

URI	<code>/api/v1/configuration/connector</code>
Method	<code>PUT</code>
Request	<code>{description}</code>

Response	<code>{ha, description}</code>
Errors	INVALID_REQUEST
Roles	Administrator

- **Request Properties:**
`description`: A string; use an empty string to remove the description.
- **Response Properties:**
 - `ha`: Cloud Connector instance assigned to a high-availability role. Its value is either the string *master* or *shadow*.
 - `description`: Description of the Cloud Connector.
- **Errors:**
INVALID_REQUEST ((400): The value of `description` is not a JSON string.

Note

`null` is not a JSON string.

1.2.2.5.2 High Availability Settings

Read and edit the high availability settings of a Cloud Connector instance via API.

When installing a Cloud Connector instance, you usually define its high availability role (master or shadow instance) during initial configuration, see [Change your Password and Choose Installation Type \[page 390\]](#).

If the high availability role was not defined before, you can set the master or shadow role via this API.

If a shadow instance is connected to the master, this API also lets you switch the roles: the master instance requests the shadow instance to take over the master role, and then takes the shadow role itself.

Note

Editing the high availability settings is only allowed on the master instance, and supports only **shadow** as input.

Get High Availability Settings

URI	<code>/api/v1/configuration/connector/haRole</code>
-----	---

Method	GET
Request	
Response	"<HA role>"
Errors	
Roles	Administrator, Display, Support

Example

```
curl -i -k -u Administrator:<password> https://localhost:8443/api/v1/
configuration/connector/haRole
```

Set High Availability Settings

Use this API if you want to set the role of a fresh installation (no role assigned yet).

As of version 2.12.0, this API also allows to switch the roles if a shadow instance is connected to the master. In this case, the API is only allowed on the master instance and supports only the value **shadow** as input. The master instance requests the shadow instance to take over the master role and then assumes the shadow role itself.

URI	/api/v1/configuration/connector/haRole
Method	POST
Request	"master" or "shadow"
Response	
Errors	ILLEGAL_STATE, INVALID_REQUEST
Roles	Administrator

Errors:

- **INVALID_REQUEST** (400): If a high-availability role other than "master" or "shadow" is supplied.
- **ILLEGAL_STATE** (409): If changing the high-availability role is not possible; changing the role is only possible if no role has been assigned, or if the current role is master and a shadow system is connected.

Example

```
curl -i -k -H 'Content-Type: application/json' -d '"shadow"' -u Administrator:<password> -X POST https://localhost:8443/api/v1/configuration/connector/haRole
```

Related Information

[Master Instance Configuration \[page 487\]](#)

[Shadow Instance Configuration \[page 491\]](#)

1.2.2.5.2.1 Master Instance Configuration

Read and edit the high availability settings for a Cloud Connector master instance via API.

Note

The APIs below are available as of Cloud Connector version 2.13.0.

Restriction

These APIs are only permitted on a Cloud Connector master instance. The shadow instance rejects the requests with error code *400 – Invalid Request*.

Get Configuration

URI	/api/v1/configuration/connector/ha/master/config
Method	GET
Request	
Response	{haEnabled, haPort, allowedShadowHost}
Errors	

Response Properties:

- `haEnabled`: Boolean value that indicates whether or not a shadow system is allowed to connect.
- `haPort`: Port on which the shadow instance can connect (restart required after change).
- `allowedShadowHost`: Name of the shadow host (a string) that is allowed to connect; an empty string signifies that any host is allowed to connect as shadow.

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/configuration/connector/ha/master/config
```

Set Configuration

URI	/api/v1/configuration/connector/ha/master/config
Method	PUT
Request	{haEnabled, haPort, allowedShadowHost}
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Response Properties:

- `haEnabled`: Boolean value that indicates whether or not a shadow system is allowed to connect.
- `haPort`: Port on which the shadow instance can connect (restart required after change).
- `allowedShadowHost`: Name of the shadow host (a string) that is allowed to connect. An empty string means that any host is allowed to connect as shadow.

Errors:

- `INVALID_REQUEST` (400): if the name of the shadow host is not a valid host name

Example

```
curl -i -k -u <user>:<password> -X PUT -H 'Content-Type: application/json'
-d '{"haEnabled":true}' https://<host>:<port>/api/v1/configuration/connector/ha/
master/config
```

Get State

URI	/api/v1/configuration/connector/ha/ master/state
Method	GET
Request	
Response	{state, shadowHost}
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `state`: One of the following strings: ALONE, BINDING, CONNECTED or BROKEN.
- `shadowHost`: Connected shadow host (a string)

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/
configuration/connector/ha/master/state
```

Set State

URI	/api/v1/configuration/connector/ha/ master/state
Method	POST

Request	
Response	{op}
Errors	ILLEGAL_STATE, INVALID_REQUEST
Roles	Administrator

Request Properties:

The value of property `op` is one of the following strings:

- `SWITCH`: Switch roles with shadow
- `FORCE_SWITCH`: Take over the shadow role, even if shadow instance does not respond.

Errors:

- `INVALID_REQUEST` (400): Value of property `op` is neither `SWITCH` nor `FORCE_SWITCH`.
- `ILLEGAL_STATE` (409): Master system is in a state that does not permit the requested operation.

Example

```
curl -i -k -u <user>:<password> -X POST -H 'Content-Type: application/json' -d
'{"op": "SWITCH"}' https://<host>:<port>/api/v1/configuration/connector/ha/master/
state
```

Reset

A successful call to this API restores default values for all settings related to high availability on the master side.

⚠ Caution

Do not perform this call if the shadow is connected to a master.

URI	/api/v1/configuration/connector/ha/master/state
Method	<i>DELETE</i>
Request	
Response	204 on success

Errors	ILLEGAL_STATE
Roles	Administrator

Errors:

- `ILLEGAL_STATE` (409): A shadow instance is connected to the master.

Example

```
curl -i -k -u <user>:<password> -X DELETE https://<host>:<port>/api/v1/
configuration/connector/ha/master/state
```

1.2.2.5.2.2 Shadow Instance Configuration

Read and edit the configuration settings for a Cloud Connector shadow instance via API (available as of Cloud Connector version 2.12.0, or, where mentioned, as of version 2.13.0).

Note

The APIs below are only permitted on a Cloud Connector shadow instance. The master instance will reject the requests with error code 403 - `FORBIDDEN_REQUEST`.

Get Configuration

URI	<code>/api/v1/configuration/connector/ha/shadow/config</code>
Method	<code>GET</code>
Request	
Response	<pre>{masterHost, masterPort, ownHost, haPort, checkIntervalInSeconds, takeoverDelayInSeconds, connectTimeoutInMillis, requestTimeoutInMillis}</pre>
Errors	

Response Properties:

- `masterHost`: Host name of the master instance (string)
- `masterPort`: Port of the master instance (string or number)
- `ownHost`: Host name of the shadow instance (string)
- `haPort`: Own port for HA communication (restart required after change).
- `checkIntervalInSeconds`: Time between two health checks against the master instance (number)
- `takeoverDelayInSeconds`: The time a master instance may stay unreachable before the shadow instance takes over (number)
- `connectTimeoutInMillis`: Timeout for connection attempts between shadow and master instance (number)
- `requestTimeoutInMillis`: Timeout for requests between shadow and master instance (number)

Note

This API may take some time to fetch the own hosts from the environment.

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/
configuration/connector/ha/shadow/config
```

Set Configuration

URI	<code>/api/v1/configuration/connector/ha/shadow/config</code>
Method	<i>PUT</i>
Request	<pre>{masterPort, masterHost, ownHost, haPort, checkIntervalInSeconds, takeoverDelayInSeconds, connectTimeoutInMillis, requestTimeoutInMillis}</pre>

Response

```
{masterPort, masterHost, ownHost,
haPort, checkIntervalInSeconds,
takeoverDelayInSeconds,
connectTimeoutInMillis,
requestTimeoutInMillis}
```

Errors

Roles

Administrator

Request Properties:

- `masterHost`: Host name of the master instance (string)
- `masterPort`: Port of the master instance (string or number)
- `ownHost`: Host name of the shadow instance (string)
- `haPort`: Own port for HA communication (restart required after change).
- `checkIntervalInSeconds`: Time between two health checks against the master instance (number)
- `takeoverDelayInSeconds`: The time a master instance may stay unreachable before the shadow instance takes over (number)
- `connectTimeoutInMillis`: Timeout for connection attempts between shadow and master instance (number)
- `requestTimeoutInMillis`: Timeout for requests between shadow and master instance (number)

Response Properties:

See *Request Properties*.

Example

```
curl -i -k -u <user>:<password> -X PUT https://<host>:<port>/api/v1/
configuration/connector/ha/shadow/config
-H 'Content-Type: application/json' -d '{"masterHost":"localhost",
"masterPort":"8443", "ownHost":"localhost",
"checkIntervalInSeconds":30, "takeoverDelayInSeconds":10,
"connectTimeoutInMillis":1000,
"requestTimeoutInMillis":12000}'
```

Get State

ⓘ Note

Available as of version 2.13.0.

URI	/api/v1/configuration/connector/ha/shadow/state
Method	GET
Request	
Response	{state, ownHosts, stateMessage, masterVersions}
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `state`: Possible string values are: INITIAL, DISCONNECTED, DISCONNECTING, HANDSHAKE, INITSYNC, READY, or LOST.
- `ownHosts`: List of alternative host names for the shadow instance.
- `stateMessage`: Message providing details on the current state. This property may not always be present. Typically, this property is available if an error occurred (for example, a failed attempt to connect to the master instance).
- `masterVersions`: Overview of relevant component versions of the master system, including a flag (*property ok*) that indicates whether or not there are incompatibility issues because of differing master and shadow versions.

Note

This property is only available if the shadow instance is connected to the master instance, or if there has been a successful connection to the master system at some point in the past.

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/configuration/connector/ha/shadow/state
```

Change State

URI	/api/v1/configuration/connector/ha/shadow/state
Method	POST

Request	
Response	{op, user, password}
Errors	INVALID_REQUEST, ILLEGAL_STATE, RUNTIME_FAILURE
Roles	Administrator

Request Properties:

- `op`: String value representing the state change operation. Possible values are `CONNECT` or `DISCONNECT`.
- `user`: User for logon to the master instance
- `password`: Password for logon to the master instance

Errors:

- `INVALID_REQUEST` (400): Invalid or missing property values were supplied; this includes wrong user or password
- `ILLEGAL_STATE` (409): The requested operation cannot be executed given the current state of master and shadow instance. This typically means the master instance does not allow high availability.
- `RUNTIME_FAILURE` (500): communication error between shadow and master, e.g. handshake failure or master does not respond.

Note

The logon credentials are used for initial logon to master instance only. If a shadow instance is disconnected from its master instance, it will reconnect to the (same) master instance using a certificate. Hence, user and password can be omitted when reconnecting.

Example

```
curl -i -k -u <user>:<password> -X POST https://<host>:<port>/api/v1/
configuration/connector/ha/shadow/state
-H 'Content-Type: application/json' -d '{"op": "CONNECT",
"user": "<user on master>", "password": "<password>"}'
curl -i -k -u <user>:<password> -X POST https://<host>:<port>/api/v1/
configuration/connector/ha/shadow/state
-H 'Content-Type: application/json' -d '{"op": "DISCONNECT"}'
```

Reset

Note

Available as of version 2.13.0.

A successful call to this API deletes master host and port, and restores default values for all other settings related to a connection to the master.

Caution

Do not perform this call if the shadow is connected to a master.

URI	/api/v1/configuration/connector/ha/shadow/state
Method	DELETE
Request	
Response	
Errors	ILLEGAL_STATE
Roles	Administrator

Errors:

- ILLEGAL_STATE (409): the shadow is currently connected to a master.

Example

```
curl -i -k -u <user>:<password> -X DELETE https://<host>:<port>/api/v1/configuration/connector/ha/shadow/state
```

1.2.2.5.3 Proxy Settings

Read and edit the Cloud Connector's proxy settings via API.

Get Proxy Settings

URI	/api/v1/configuration/connector/proxy
Method	GET
Request	
Response	{host, port, user}
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `host`: the name of the proxy host (a string)
- `port`: the port of the proxy host (a string)
- `user`: the user name (a string)

Sample Code

```
curl -ik -u Administrator:<password> https://localhost:8443/api/v1/configuration/connector/proxy
```

Set Proxy Settings (Master Only)

URI	/api/v1/configuration/connector/proxy
Method	PUT
Request	{host, port, user, password}
Response	
Errors	INVALID_REQUEST, FORBIDDEN_REQUEST
Roles	Administrator

Request Properties:

- `host`: the name of the proxy host (a string)
- `port`: the port of the proxy host (a string)
- `user`: the user name (a string)
- `password`: the password (a string - optional)

Errors:

- `INVALID_REQUEST` (400): invalid values were supplied, or mandatory values are missing.
- `FORBIDDEN_REQUEST` (403): the target of the call is a shadow instance.

The following example sets empty `user` and `password`.

Sample Code

```
curl -ik -u Administrator:<password> https://localhost:8443/api/v1/
configuration/connector/proxy -X PUT -H 'Content-Type: application/json' -d
'{"host": "proxy", "port": "8080"}'
```

This request removes the proxy configuration.

Sample Code

```
curl -ik -u Administrator:<password> https://localhost:8443/api/v1/
configuration/connector/proxy -X PUT -H 'Content-Type: application/json' -d
'{'
```

Remove Proxy Settings (Master Only)

URI	<code>/api/v1/configuration/connector/proxy</code>
Method	<code>DELETE</code>
Request	
Response	204 on success
Errors	<code>FORBIDDEN_REQUEST</code>
Roles	Administrator

Errors:

- `FORBIDDEN_REQUEST` (403): the target of the call is a shadow instance.

Sample Code

```
curl -ik -u Administrator:<password> https://localhost:8443/api/v1/
configuration/connector/proxy -X DELETE
```

1.2.2.5.4 Authentication and UI Settings

Read and edit the Cloud Connector's authentication and UI settings via API.

Get Authentication Settings

URI	/api/v1/configuration/connector/authentication
Method	GET
Request	
Response	{type, configuration}
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `type`: The authentication type, which is one of the following strings: `basic` or `ldap`.
- `configuration`: The configuration of the active LDAP authentication. This property is only available if `type` is `ldap`. Its value is an object with properties that provide details on LDAP configuration.

Example

```
curl -i -k -H 'Accept:application/json'
-u Administrator:<password> -X GET https://<scchost>:8443/api/v1/configuration/
connector/authentication
```

Change Basic Authentication User

URI	/api/v1/configuration/connector/authentication/basic
Method	PUT

Request	<code>{password, user}</code>
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `password`: The current password (a string)
- `user`: The new user name (a string), overwriting the current user name.

Errors:

- `INVALID_REQUEST` (400): Current password is wrong.

Change Basic Authentication Password

URI	<code>/api/v1/configuration/connector/authentication/basic</code>
Method	<i>PUT</i>
Request	<code>{oldPassword, newPassword}</code>
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `oldPassword`: The current password, about to be changed (a string)
- `newPassword`: The new password (a string)

Errors:

- `INVALID_REQUEST` (400): Passwords are the same or current password is wrong.

Example

```
curl -i -k -H 'Content-Type:application/json' -d '{"oldPassword":"manage",
"newPassword":"test"}'
-u Administrator:manage -X PUT https://localhost:8443/api/v1/configuration/
connector/authentication/basic
```

Change LDAP Authentication

⚠ Caution

The Cloud Connector will restart if the request was successful. There is no test that confirms login will work afterwards. If you run into problems you can revert to basic authentication by executing the script `useFileUserStore` located in the root directory of your Cloud Connector installation.

URI	<code>/api/v1/configuration/connector/authentication/ldap</code>
Method	<code>PUT</code>
Request	<code>{enable, configuration}</code>
Response	204 on success
Errors	INVALID_REQUEST, INVALID_CONFIGURATION
Roles	Administrator

Request Properties:

- `enable`: Boolean flag that indicates whether or not to employ LDAP authentication.
- `configuration`: The LDAP configuration, a JSON object with the properties `{config, hosts, user, password, customAdminRole, customDisplayRole, customMonitoringRole, customSupportRole}`.
 - Property `hosts` is an array. Each element of the array defines a host, again specified through a JSON object, with the properties `{host, port, isSecure}`, accepting string, string (or number), and Boolean values, respectively.
 - All properties of the top-level object except `hosts` accept string values.
 - Properties `config` and `hosts` are mandatory. The array of hosts needs to have at least one element.
 - All other properties are optional.

Errors:

- `INVALID_REQUEST (400)`: Configuration is invalid.

- `INVALID_CONFIGURATION (409)`: LDAP server is not accessible (does not respond).

Note

In both error cases nothing is stored, and LDAP authentication is disabled.

Example

```
curl -i -k -H 'Content-Type: application/json' -u Administrator:<password>
-X PUT https://localhost:8443/api/v1/configuration/connector/authentication/
ldap -d '{"enable":true, "configuration":{"hosts":[{"host":"ldaphost",
"port":"10389", "isSecure":false}], "config":{"roleBase="\ou=groups,dc=ccc\"
roleName=\"cn\" roleSearch=\"(uniqueMember={0})\"
userBase=\"\ou=users,dc=ccc\" userSearch=\"(uid={0})\"\", \"user\":\"ldapadmin\",
\"password\":\"<ldapadminpassword>\"}}'
```

Get Description for UI Certificate

This API returns a textual description for the system certificate.

Note

Available as of version 2.13.0.

URI	<code>/api/v1/configuration/connector/ui/uiCertificate</code>
Method	<code>GET</code>
Response	<code>{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}</code>
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `subjectDN`: The subject distinguished name (a string)
- `issuer`: The issuer (a string)
- `notBeforeTimeStamp`: Timestamp of the beginning of the validity period (a UTC long number)
- `notAfterTimeStamp`: Timestamp of the end of the validity period (a UTC long number)

- `subjectAltNames`: Subject alternative names, as an array of objects with properties `type` and `value`. The value of property `type` is one of the following strings: IP, DNS, URI, or RFC822. The value of property `value` is the associated value (a string).

Note

`subjectAltNames` is not present if there are no subject alternative names.

Example

```
curl -i -k -H "Accept: application/json" -u <user>:<password> -X GET https://<host>:<port>/api/v1/configuration/connector/ui/uiCertificate
```

Create a Self-Signed UI Certificate

Note

Available as of version 2.13.0.

URI	<code>/api/v1/configuration/connector/ui/uiCertificate</code>
Method	<i>POST</i>
Request	<code>{type, keySize, subjectDN, subjectAltNames}</code>
Response	201 on success
Errors	
Roles	Administrator

Request Properties:

- `type`: The string *selfsigned*
- `keySize`: Key size, which must be either 2048 or 4096 bits. Optional property. Default value is 4096.

Note

This property is available as of version 2.15.0.

- `subjectDN`: The subject distinguished name (a string)

- `subjectAltNames`: Subject alternative names, as an array of objects with properties `type` and `value`. The value of property `type` is one of the following strings: IP, DNS, URI, or RFC822. The value of property `value` is the associated value (a string). This property is optional.

The UI certificate created this way has a validity of 1 year.

Example

```
curl -k -u Administrator:<password> -X POST -H "Content-Type: application/json"
--data '{"type":"selfsigned", "subjectDN":"CN=me"}'
https://<host>:<port>/api/v1/configuration/connector/ui/uiCertificate
```

Create a Certificate Signing Request for a UI Certificate

Note

Available as of version 2.13.0.

URI	/api/v1/configuration/connector/ui/uiCertificate
Method	POST
Request	{type, keySize, subjectDN, subjectAltNames}
Response	PEM-encoded certificate request
Errors	
Roles	Administrator

Request Properties:

- `type`: The string `csr`
- `keySize`: Key size, which must be either 2048 or 4096 bits. Optional property. Default value is 4096.

Note

This property is available as of version 2.15.0.

- `subjectDN`: The subject distinguished name (a string)
- `subjectAltNames`: Subject alternative names, as an array of objects with properties `type` and `value`. The value of property `type` is one of the following strings: IP, DNS, URI, or RFC822. The value of property `value` is the associated value (a string). This property is optional.

Example

```
curl -k -u Administrator:<password> -X POST -H "Content-Type: application/json"
--data '{"type":"csr", "subjectDN":"CN=me"}'
https://<host>:<port>/api/v1/configuration/connector/ui/uiCertificate -o
csr.pem
```

Upload a Signed Certificate Chain as UI Certificate

Note

Available as of version 2.13.0.

URI	/api/v1/configuration/connector/ui/uiCertificate
Method	PATCH
Request	multipart/form-data with the following form parameter: signedCertificate
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- signedCertificate: The signed certificate and CA certificate chain (PEM-encoded)

Errors:

- INVALID_REQUEST (400): The certificate chain provided does not match the most recent certificate request, or it is not a certificate chain in the proper format (PEM-encoded).

Example

```
curl -i -k -u <user>:<password> -X PATCH -F signedCertificate=@<signedchain.pem>
https://<host>:<port>/api/v1/configuration/connector/ui/uiCertificate
```


Example

For test purposes, you can sign the certificate signing request with keytool.

```
keytool -genkeypair -keyalg RSA -keysize 1024 -alias mykey -dname "cn=very
trusted, c=test" -validity 365 -keystore ca.ks -keypass testit -storepass testit
keytool -gencert -rfc -infile csr.pem -outfile signedcsr.pem -alias mykey
-keystore ca.ks -keypass testit -storepass testit
keytool -exportcert -rfc -file ca.pem -alias mykey -keystore ca.ks -keypass
testit -storepass testit
cat signedcsr.pem ca.pem > signedchain.pem
```

Upload a PKCS#12 Certificate as UI Certificate

Note

Available as of version 2.13.0.

URI	/api/v1/configuration/connector/ui/ uiCertificat
Method	PUT
Request	multipart/form-data with the following form parameters: pkcs12 password keyPassword
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- `pkcs12`: Contents of PKCS#12 file
- `password`: Password for decrypting the PKCS#12 file
- `keyPassword`: Optional password for the private key

Errors:

- `INVALID_REQUEST (400)`: Contents of PKCS#12 or password are invalid

Note

keyPassword is optional. If missing, password is used to decrypt the pkcs#12 file and the private key.

Example

```
curl -i -k -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/ui/uiCertificate -X PUT -F 'password=<p12Password>' -F pkcs12=@<p12file>
```

Example

For test purposes, you can create an own self-signed pkcs#12 certificate with keytool.

```
keytool -genkeypair -alias key -keyalg RSA -keysize 2048 -validity 365 -keypass test20 -keystore test.p12 -storepass test20 -storetype PKCS12 -dname 'CN=test'
```

Get List of Available Cipher Suites for UI

Note

Available as of version 2.15.0.

Returns a list of available cipher suites (as per JVM).

URI	/api/v1/configuration/availableCipherSuites
Method	GET
Request	
Response	[name, ...]
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response Properties:

List of:

- name: Name of a cipher suite (a string).

Note

name is case-sensitive.

Example

```
curl -i -k -H "Accept: application/json" -u <user>:<password> -X GET https://<host>:<port>/api/v1/configuration/availableCipherSuites
```

Get List of Enabled Cipher Suites for UI

Note

Available as of version 2.15.0.

Returns a list of enabled cipher suites.

URI	/api/v1/configuration/connector/ui/cipherSuites
Method	GET
Request	
Response	[name, ...]
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response Properties:

List of:

- name: Name of a cipher suite (a string).

Note

name is case-sensitive.

Example

```
curl -i -k -H "Accept: application/json" -u <user>:<password> -X GET https://<host>:<port>/api/v1/configuration/connector/ui/uiCipherSuites
```

Set List of Enabled Cipher Suites for UI

Note

Available as of version 2.15.0.

URI	/api/v1/configuration/connector/ui/cipherSuites
Method	POST
Request	[name, ...]
Response	204 on success
Errors	
Roles	Administrator

Request Properties:

List of:

- name: Name of a cipher suite (a string).

Note

name is case-sensitive.

Example

```
curl -i -k -H "Content-Type: application/json" -u <user>:<password> --  
data ["'TLS_AES_128_GCM_SHA256', 'TLS_AES_256_GCM_SHA384']" -X POST https://  
<host>:<port>/api/v1/configuration/connector/ui/cipherSuites
```

Enable default Cipher Suites for UI

Note

Available as of version 2.15.0.

Revert changes and enable the default list of cipher suites.

URI	/api/v1/configuration/connector/ui/ cipherSuites
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Response Properties:

- name: Name of cipher suite (a string).

Note

name is case-sensitive.

Example

```
curl -i -k -u <user>:<password> -X DELETE https://<host>:<port>/api/v1/  
configuration/connector/ui/cipherSuites
```

1.2.2.5.5 Certificate Management for Backend Communication

Manage a CA certificate for principal propagation or a system certificate via API.

Note

The APIs below are available as of Cloud Connector version 2.13.

There are two similar sets of APIs for system certificate and CA certificate for principal propagation.

Note

Some of the APIs list a parameter `subjectAltNames` (*subject alternative names* or SAN) for the request or response object. This parameter is an array of objects with the following properties:

- `type`: one of the strings *DNS*, *URI*, *IP*, or *RFC822*.
- `value`: the value associated with the chosen type.

- [CA Certificate for Principal Propagation: APIs \[page 511\]](#)
- [System Certificate: APIs \[page 517\]](#)
- [Truststore CA Certificates \[page 524\]](#)

1.2.2.5.5.1 CA Certificate for Principal Propagation: APIs

Manage a CA certificate for principal propagation via API.

Note

The APIs below are available as of Cloud Connector version 2.13.0.

Get Description for a CA Certificate for Principal Propagation

URI	<code>/api/v1/configuration/connector/onPremise/ppCaCertificate</code>
Method	<code>GET</code>
Header	<code>Accept: application/json</code>

Response	<code>{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}</code>
Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response Properties:

- `subjectDN`: the subject distinguished name (a string)
- `issuer`: the issuer (a string)
- `notBeforeTimeStamp`: timestamp of the beginning of the validity period (a UTC long number)
- `notAfterTimeStamp`: timestamp of the end of the validity period (a UTC long number)
- `subjectAltNames`: subject alternative names (see [Certificate Management for Backend Communication \[page 511\]](#) for details).

Errors:

- NOT_FOUND (404): there is no CA certificate for principal propagation.

Note

`subjectAltNames` is not present if there are no subject alternative names.

Example

```
curl -i -k -H "Accept: application/json" -u <user>:<password> -X GET https://<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate
```

Get Binary Content of a CA Certificate for Principal Propagation

URI	<code>/api/v1/configuration/connector/onPremise/ppCaCertificate</code>
Method	<i>GET</i>
Header	Accept: application/pkix-cert
Response	Binary data of the certificate.

Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response:

- **Success:** the binary data of the certificate; you can verify the downloaded certificate by storing it in file `ppca.crt`, for instance, and then running

```
keytool -printcert -file ppca.crt
```

- **Failure:** an error in the usual JSON format; the content type of the response is `application/json` in this case.

Errors:

- `NOT_FOUND (404)`: there is no CA certificate for principal propagation.

Example

```
curl -k -H "Accept: application/pkix-cert" -u <user>:<password> -X GET
--output sys.crt https://<host>:<port>/api/v1/configuration/connector/onPremise/
ppCaCertificate
```

Create a Self-Signed CA Certificate for Principal Propagation (Master Only)

URI	<code>/api/v1/configuration/connector/onPremise/ppCaCertificate</code>
Method	<code>POST</code>
Request	<code>{type, keySize, subjectDN, subjectAltNames}</code>
Response	201 on success
Errors	
Roles	Administrator

Request Properties:

- `type`: the string `selfsigned`
- `keySize`: Key size, which must be either 2048 or 4096 bits. Optional property. Default value is 4096.

Note

This property is available as of version 2.15.0.

- `subjectDN`: the subject distinguished name (a string)
- `subjectAltNames`: subject alternative names (see [Certificate Management for Backend Communication \[page 511\]](#) for details). This property is optional.

The certificate created this way has a validity of 1 year.

Example

```
curl -i -k -H "Accept: application/json" -H "Content-Type: application/json" -u <user>:<password> -X POST --data '{"type":"selfsigned", "subjectDN":"CN=me"}' https://<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate
```

Create a Certificate Signing Request for a CA Certificate for Principal Propagation (Master Only)

URI	/api/v1/configuration/connector/onPremise/ppCaCertificate
Method	POST
Request	{type, keySize, subjectDN, subjectAltNames}
Response	PEM-encoded certificate request
Errors	
Roles	Administrator

Request Properties:

- `type`: the string *selfsigned*
- `keySize`: Key size, which must be either 2048 or 4096 bits. Optional property. Default value is 4096.

Note

This property is available as of version 2.15.0.

- `subjectDN`: the subject distinguished name (a string)

- `subjectAltNames`: subject alternative names (see [Certificate Management for Backend Communication \[page 511\]](#) for details). This property is optional.

Example

```
curl -k -u <user>:<password> -X POST -H "Content-Type: application/json" --data
'{"type": "csr", "subjectDN": "CN=me"}'
https://<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate
-o csr.pem
```

Upload a Signed Certificate Chain as CA Certificate for Principal Propagation (Master Only)

URI	/api/v1/configuration/connector/ onPremise/ppCaCertificate
Method	<i>PATCH</i>
Request	multipart/form-data with the following parameter: signedCertificate.
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `signedCertificate`: the signed certificate and CA certificate chain (PEM-encoded)

Errors:

- `INVALID_REQUEST (400)`: the certificate chain provided does not match the most recent certificate request, or it is not a certificate chain in the correct format (PEM-encoded).

Example

```
curl -i -k -u <user>:<password> -X PATCH -F signedCertificate=@<signedchain.pem>
https://<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate
```

Example: Sign The Certificate Signing Request

```
keytool -genkeypair -keyalg RSA -keysize 1024 -alias mykey -dname "cn=very
trusted, c=test" -validity 365 -keystore ca.ks -keypass testit -storepass testit
keytool -gencert -rfc -infile csr.pem -outfile signedcsr.pem -alias mykey
-keystore ca.ks -keypass testit -storepass testit
keytool -exportcert -rfc -file ca.pem -alias mykey -keystore ca.ks -keypass
testit -storepass testit
cat signedcsr.pem ca.pem > signedchain.pem
```

Upload a PKCS#12 Certificate as CA Certificate for Principal Propagation (Master Only)

URI	/api/v1/configuration/connector/ onPremise/ppCaCertificate
Method	PUT
Request	Multipart/form-data with the following form parameters: pkcs12, password, keyPassword
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- `pkcs12`: contents of PKCS#12 file
- `password`: password for decrypting PKCS#12 file
- `keyPassword`: optional password for the private key

Errors:

- `INVALID_REQUEST (400)`: contents of PKCS#12 or password are invalid

Note

`keyPassword` is optional. If it is missing, `password` is used to decrypt the `pkcs#12` file and the private key.

Example

```
curl -i -k -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate -X PUT -F 'password=<p21Password>' -F pkcs12=@<p12file>
```

Create an own self-signed pkcs#12 certificate for tests with:

```
keytool -genkeypair -alias key -keyalg RSA -keysize 2048 -validity 365 -keypass test20 -keystore test.p12 -storepass test20 -storetype PKCS12 -dname 'CN=test'
```

Delete a CA Certificate for Principal Propagation (Master Only)

URI	/api/v1/configuration/connector/onPremise/ppCaCertificate
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator

Errors:

- NOT_FOUND (404): there is no CA certificate for principal propagation.

Example

```
curl -k -H "Accept: application/json" -u <user>:<password> --request DELETE https://localhost:8443/api/v1/configuration/connector/onPremise/ppCaCertificate
```

1.2.2.5.5.2 System Certificate: APIs

Manage a system certificate via API.

Note

The APIs below are available as of Cloud Connector version 2.13.0.

Get Description for a System Certificate

URI	<code>/api/v1/configuration/connector/onPremise/systemCertificate</code>
Method	<code>GET</code>
Header	Accept: application/json
Response	<pre>{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}</pre>
Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response Properties:

- `subjectDN`: the subject distinguished name (a string)
- `issuer`: the issuer (a string)
- `notBeforeTimeStamp`: timestamp of the beginning of the validity period (a UTC long number)
- `notAfterTimeStamp`: timestamp of the end of the validity period (a UTC long number)
- `subjectAltNames`: subject alternative names.

Errors:

- `NOT_FOUND` (404): there is no system certificate.

Note

`subjectAltNames` is not present if there are no subject alternative names.

Example

```
curl -i -k -H "Accept: application/json" -u <user>:<password> -X GET https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate
```

Get Binary Content of a System Certificate

URI	/api/v1/configuration/connector/onPremise/systemCertificate
Method	GET
Header	Accept: application/pkix-cert
Response	Binary data of the certificate.
Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response:

- **Success:** the binary data of the certificate; you can verify the downloaded certificate by storing it in file `sys.crt`, for instance, and then running

```
keytool -printcert -file sys.crt
```

- **Failure:** an error in the usual JSON format; the content type of the response is `application/json` in this case.

Errors:

- `NOT_FOUND` (404): there is no system certificate.

Example

```
curl -i -k -H "Accept: application/pkix-cert" -u <user>:<password> -X GET --output sys.crt https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate
```

Create a Self-Signed System Certificate (Master Only)

URI	/api/v1/configuration/connector/onPremise/systemCertificate
Method	POST
Header	CONTENT_TYPE: application/json

Request	{type, keySize, subjectDN}
Response	201 on success
Errors	
Roles	Administrator

Request Properties:

- type: the string *selfsigned*
- keySize: Key size, which must be either 2048 or 4096 bits. Optional property. Default value is 4096.

Note

This property is available as of version 2.15.0.

- subjectDN: the subject distinguished name (a string)
- subjectAltNames: subject alternative names. This property is optional. As of version 2.16.1, the value is ignored. SAN on system certificate had never any effect.

The certificate created this way has a validity of 1 year.

Example

```
curl -i -k -H "Accept: application/json" -H "Content-Type: application/json" -u <user>:<password> -X POST --data '{"type":"selfsigned", "subjectDN":"CN=me"}' https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate
```

Create a Certificate Signing Request for a System Certificate (Master Only)

URI	/api/v1/configuration/connector/onPremise/systemCertificate
Method	POST
Header	CONTENT_TYPE: application/json
Request	{type, keySize, subjectDN}
Response	PEM-encoded certificate request

Errors

Roles

Administrator

Request Properties:

- `type`: the string `csr`
- `keySize`: Key size, which must be either 2048 or 4096 bits. Optional property. Default value is 4096.

Note

This property is available as of version 2.15.0.

- `subjectDN`: the subject distinguished name (a string)
- `subjectAltNames`: subject alternative names. This property is optional. As of version 2.16.1, the value is ignored. SAN on system certificate had never any effect.

Example

```
curl -k -u <user>:<password> -X POST -H "Content-Type: application/json" --data '{ "type": "csr", "subjectDN": "CN=me" }' https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate -o csr.pem
```

Upload a Signed Certificate Chain as System Certificate (Master Only)

URI	<code>/api/v1/configuration/connector/onPremise/systemCertificate</code>
Method	<code>PATCH</code>
Request	multipart/form-data with the following parameter: <code>signedCertificate</code> .
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `signedCertificate`: the signed certificate and CA certificate chain (PEM-encoded)

Errors:

- `INVALID_REQUEST (400)`: the certificate chain provided does not match the most recent certificate request, or it is not a certificate chain in the correct format (PEM-encoded).

Example

```
curl -i -k -u <user>:<password> -X PATCH -F signedCertificate=@<signedchain.pem> https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate
```

For test purposes, you can sign the certificate signing request with keytool:

```
keytool -genkeypair -keyalg RSA -keysize 1024 -alias mykey -dname "cn=very trusted, c=test" -validity 365 -keystore ca.ks -keypass testit -storepass testit
keytool -gencert -rfc -infile csr.pem -outfile signedcsr.pem -alias mykey -keystore ca.ks -keypass testit -storepass testit
keytool -exportcert -rfc -file ca.pem -alias mykey -keystore ca.ks -keypass testit -storepass testit
cat signedcsr.pem ca.pem > signedchain.pem
```

Upload a PKCS#12 Certificate as System Certificate (Master Only)

URI	/api/v1/configuration/connector/onPremise/systemCertificate
Method	PUT
Request	Multipart/form-data with the following form parameters: pkcs12, password, keyPassword
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- `pkcs12`: contents of PKCS#12 file
- `password`: password for decrypting PKCS#12 file
- `keyPassword`: optional password for the private key

Errors:

- `INVALID_REQUEST (400)`: contents of PKCS#12 or password are invalid

Note

`keyPassword` is optional. If it is missing, `password` is used to decrypt the pkcs#12 file and the private key.

Example

```
curl -i -k -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate -X PUT -F password=<p21Password> -F pkcs12=@<p12file>
```

Create an own self-signed pkcs#12 certificate for tests with:

```
keytool -genkeypair -alias key -keyalg RSA -keysize 2048 -validity 365 -keypass test20 -keystore test.p12 -storepass test20 -storetype PKCS12 -dname 'CN=test'
```

Delete a System Certificate (Master Only)

URI	/api/v1/configuration/connector/onPremise/systemCertificate
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator

Errors:

- `NOT_FOUND (404)`: there is no system certificate.

Example

```
curl -k -H "Accept: application/json" -u <user>:<password> --request DELETE https://localhost:8443/api/v1/configuration/connector/onPremise/systemCertificate
```

1.2.2.5.5.3 Truststore CA Certificates

Manage CA certificates in the truststore via API.

Get Truststore Configuration

Note

This API is available as of Cloud Connector version 2.15.0.

URI	/api/v1/configuration/connector/onPremise/truststore
Method	GET
Header	
Response	{trustAllBackends, trustedBackends}
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response Properties:

- `trustAllBackends`: flag (boolean) indicating whether all backends are trusted (**true**), or only the backends represented by certificates are trusted (**false**).
- `trustedBackendSystems`: list (array) of certificates representing the trusted backends. Each certificate is specified through an object with the following properties:

Certificate Properties:

- `alias`: alias of CA certificate (a string).
- `subjectDN`: subject DN (a string).
- `issuer`: the issuer (a string).
- `notAfterTimeStamp`: (a UTC long number).

Note

No HTTP requests can be executed if `trustAllBackends` is **false** and the certificate list as per `trustedBackendSystems` is empty.

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/
configuration/connector/onPremise/truststoreConfiguration
```

Change Truststore Configuration (Master Only)

Note

This API is available as of Cloud Connector version 2.15.0.

URI	/api/v1/configuration/connector/ onPremise/truststore
Method	<i>PATCH</i>
Header	Accept: application/json
Request	{trustAllBackends}
Response	204 on success
Errors	
Roles	Administrator

Request Properties:

- `trustAllBackends`: flag (boolean) indicating whether all backends are trusted (**true**), or only the backends represented by certificates are trusted (**false**)

Caution

We discourage trusting all backends and recommend that you trust only specific backends represented by their respective certificates.

Example

```
curl -k -u <user>:<password> -X PATCH -H "Accept: application/json"
-d '{"trustAllBackends": true}' https://<host>:<port>/api/v1/configuration/
connector/onPremise/truststore
```

Add a CA Certificate to Truststore (Master Only)

Note

This API is available as of Cloud Connector version 2.15.0.

URI	/api/v1/configuration/connector/ onPremise/truststore/certificates
Method	POST
Header	CONTENT_TYPE: multipart/form-data
Request	multipart/form-data with the following form parameter: certificate
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `certificate`: certificate in DER or PEM format.

Errors:

- `INVALID_REQUEST`: if the uploaded certificate is already available.

Example

```
curl -k -u <user>:<password> -X POST -F 'certificate=@<file>' https://
<host>:<port>/api/v1/configuration/connector/onPremise/truststore/certificates
```

Delete a CA Certificate From Truststore (Master Only)

Note

This API is available as of Cloud Connector version 2.15.0.

URI	<code>/api/v1/configuration/connector/onPremise/truststore/certificates/<alias></code>
Method	<code>DELETE</code>
Header	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator

Errors:

- NOT_FOUND: if the certificate with the given alias is not available.

Example

```
curl -k -H "Accept: application/json" -u <user>:<password> --request DELETE https://localhost:8443/api/v1/configuration/connector/onPremise/truststore/certificates/<alias>
```

Delete All CA Certificates From Truststore (Master Only)

Note

This API is available as of Cloud Connector version 2.14.1.

URI	<code>/api/v1/configuration/connector/onPremise/truststore/certificates</code>
-----	--

Method	<i>DELETE</i>
Header	
Response	204 on success
Errors	
Roles	Administrator

Example

```
curl -k -H "Accept: application/json" -u <user>:<password> --
request DELETE https://localhost:8443/api/v1/configuration/connector/onPremise/
truststore/certificates
```

1.2.2.5.6 Solution Management Configuration

Manage the Cloud Connector's solution management configuration via API.

Get Solution Management Configuration

URI	/api/v1/configuration/connector/ solutionManagement
Method	<i>GET</i>
Request	
Response	{hostAgentPath, isEnabled, dsrEnabled}
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `isEnabled`: flag indicating if reporting to solution management is active.
- `hostAgentPath`: path for host agent executable.

- `dsrEnabled`: indicating if DSR reporting is active.

Example

```
curl -ik -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/solutionManagement
```

Set Solution Management Configuration and Turn On Reporting

This API turns on the integration with the Solution Manager. The prerequisite is an available Host Agent. You can specify a path to the Host Agent executable, if you don't use the default path.

URI	/api/v1/configuration/connector/solutionManagement
Method	POST
Request	{hostAgentPath, dsrEnabled}
Response	
Errors	
Roles	Administrator, Support

Response Properties:

- `hostAgentPath`: path for host agent executable (string, optional).
- `dsrEnabled`: flag indicating if DSR reporting is active (boolean, optional)-

Example

```
curl -ik -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/solutionManagement -X POST
```


or, if configuration has to be changed:

```
curl -ik -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/solutionManagement -X POST -d "{\"hostAgentPath\":\"new/path/to/hostAgent\"}" -H "Content-Type:application/json"
```

Turn Off Solution Management Reporting

URI	/api/v1/configuration/connector/solutionManagement
Method	<i>DELETE</i>
Request	
Response	
Errors	
Roles	Administrator, Support

Example

```
curl -ik -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/solutionManagement -X DELETE
```

Download Current LMDB XML Report

Generates a zip file containing the registration file for the solution management LMDB (Landscape Management Database).

Note

Available as of Cloud Connector version 2.12.0.

URI	/api/v1/configuration/connector/ solutionManagement/registrationFile
Method	GET
Request	
Response	
Errors	
Roles	Administrator, Support

1.2.2.5.7 Backup

Manage the Cloud Connector's configuration backup via API.

Create Backup Configuration

URI	/api/v1/configuration/backup
Method	POST
Request	{password}
Response	ZIP archive (content type application/zip)
Errors	INVALID_REQUEST, FORBIDDEN
Roles	Administrator

Request Properties:

- `password`: password used to encrypt sensitive data.

Errors:

- `INVALID_REQUEST` (400): if no password was provided.
- `FORBIDDEN` (403): if the instance role is *shadow*.

Note

Only sensitive data in the backup are encrypted with an arbitrary password of your choice. The password is required for the restore operation. The returned ZIP archive itself is not password-protected.

Sample Code

```
curl -k -u Administrator:<password> https://<host>:<port>/api/v1/
configuration/backup -X POST -H 'Content-Type: application/json' -d
"{\"password\": \"<password>\"}" -o <backupfile>.zip
```

Restore Backup Configuration

Caution

A successful request triggers a restart of the Cloud Connector.

URI	/api/v1/configuration/backup
Method	PUT
Request	multipart/form-data with the following form parameters: backup, password.
Response	204 on success
Errors	INVALID_REQUEST, FORBIDDEN, INTERNAL_SERVER_ERROR
Roles	Administrator

Request Properties:

- `backup`: a backup file (produced through POST request).
- `password`: password chosen when creating the backup.

Errors:

- `INVALID_REQUEST` (400): if invalid or no backup file, or incorrect or missing password was provided.
- `FORBIDDEN` (403): if the instance role is `shadow`.
- `INTERNAL_SERVER_ERROR` (500): if an error happened during restore.

Note

Since this API uses a multipart request, it requires a multipart request header.

Sample Code

```
curl -k -u Administrator:<password> https://<host>:<port>/api/v1/
configuration/backup -X PUT -F 'password=<password>' -F
backup=@<backupfile>.zip
```

1.2.2.5.8 Subaccount

Manage the Cloud Connector's subaccount settings via API.

Operations

Subaccount

- [Get Subaccounts \[page 534\]](#)
- [Create Subaccount \(Master Only\) \[page 534\]](#)
- [Delete Subaccount \(Master Only\) \[page 535\]](#)
- [Edit Subaccount \(Master Only\) \[page 536\]](#)
- [Connect/Disconnect Subaccount \(Master Only\) \[page 537\]](#)
- [Refresh Subaccount Certificate \(Master Only\) \[page 537\]](#)
- [Get Subaccount Configuration \[page 538\]](#)

Recovery Subaccount

⚠ Caution

This feature is deprecated.

Due to the discontinuation of the *Enhanced Disaster Recovery Service*, the related functionality in the Cloud Connector has been dropped as of version 2.16.

For more information, see [What's New for SAP Business Technology Platform](#).

- [Create Recovery Subaccount \(Master Only\) \[page 539\]](#)
- [Delete Recovery Subaccount \(Master Only\) \[page 539\]](#)
- [Refresh Certificate of Recovery Subaccount \(Master Only\) \[page 539\]](#)
- [Activate/Deactivate Recovery Subaccount \(Master Only\) \[page 540\]](#)
- [Takeover \(Master Only\) \[page 540\]](#)

Trust

- [Synchronize Trust List \(Master Only\) \[page 540\]](#)
- [Get IDP Trust List \[page 541\]](#)
- [Get Application Trust List \[page 542\]](#)
- [Enable Or Disable Trust \(Master Only\) \[page 542\]](#)
- [Get IDP Trust Properties \[page 543\]](#)
- [Get Application Trust Properties \[page 544\]](#)

Get Subaccounts

URI	/api/v1/configuration/subaccounts
Method	GET
Request	
Response	<pre>[{regionHost, subaccount, locationID}]</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects with the following properties:

- `regionHost`: region hosts (a string).
- `subaccount`: subaccount name (a string).
- `locationID`: location identifier for the Cloud Connector instance (a string); this property is not available if the default location ID is in use.

Back to [Operations \[page 533\]](#)

Create Subaccount (Master Only)

Creates and connects a subaccount.

URI	/api/v1/configuration/subaccounts
Method	POST
Request	<pre>{regionHost, subaccount, cloudUser, cloudPassword, locationID, displayName, description}</pre>
Response	201, created subaccount entity: <pre>{regionHost, subaccount, locationID, displayName, description, tunnel}</pre>
Errors	INVALID_REQUEST, INVALID_CONFIGURATION

Request Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).
- `cloudUser`: user for the specified subaccount and region host.
- `cloudPassword`: password for the cloud user.
- `locationID`: location identifier for the Cloud Connector instance (a string; optional).
- `displayName`: display name of the subaccount (a string; optional).
- `description`: subaccount description (a string; optional).

Response Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).
- `locationID`: location ID (a string); this property is not available if the default location ID is in use.
- `displayName`: display name of the subaccount (a string); this property is not available if there is no specified display name.
- `description`: subaccount description (a string); this property is not available if there is no description.
- `tunnel`: object outlining the current state of the tunnel.

Errors:

- `INVALID_REQUEST` (400): one or more mandatory parameters are missing or invalid.
- `INVALID_CONFIGURATION` (409): the subaccount already exists.

Back to [Operations \[page 533\]](#)

Delete Subaccount (Master Only)

URI	<code>/api/v1/configuration/subaccounts/<regionHost>/<subaccount></code>
Method	<code>DELETE</code>
Request	
Response	204 on success
Errors	NOT_FOUND, ILLEGAL_STATE
Roles	Administrator, Subaccount Administrator

Errors:

- `NOT_FOUND` (404): subaccount does not exist (in the specified region).
- `ILLEGAL_STATE` (409): there is at least one session that has access to the subaccount.

Back to [Operations \[page 533\]](#)

Edit Subaccount (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount></code>
Method	<code>PUT</code>
Request	<code>{locationID, displayName, description}</code>
Response	<code>{regionHost, subaccount, locationID, displayName, description, tunnel}</code>
Errors	<code>NOT_FOUND</code>
Roles	Administrator, Subaccount Administrator

Request Properties:

- `locationID`: location identifier for the Cloud Connector instance (a string; optional); if this parameter is not supplied the location ID will not change. Revert to the default location ID by supplying the empty string.
- `displayName`: subaccount display name (a string; optional); if this parameter is not supplied the display name will not change. Clear the display name by using an empty string.
- `description`: subaccount description (a string; optional); if this parameter is not supplied the description will not change. Clear the description by using an empty string.

Response Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).
- `locationID`: location identifier for the Cloud Connector instance (a string); this property is not available if the default location ID is in use.
- `displayName`: display name of the subaccount (a string); this property is not available if there is no specified display name.
- `description`: subaccount description (a string); this property is not available if there is no description.
- `tunnel`: object outlining the current state of the tunnel.

Errors:

- NOT_FOUND (404): subaccount does not exist (in the specified region).

Back to [Operations \[page 533\]](#)

Connect/Disconnect Subaccount (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/state</code>
Method	<i>PUT</i>
Request	<code>{connected}</code>
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request Properties:

- `connected`: a Boolean value indicating whether the subaccount should be connected (true) or disconnected (false).

Back to [Operations \[page 533\]](#)

Refresh Subaccount Certificate (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/validity</code>
Method	<i>POST</i>
Request	<code>{user, password}</code>
Response	<code>{regionHost, subaccount, locationID, displayName, description, tunnel}</code>

Errors

Roles

Administrator, Subaccount Administrator

Request Properties:

- `user`: user for the specified region host and subaccount.
- `password`: password for the (cloud) user.

Response Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).
- `locationID`: location identifier for the Cloud Connector instance (a string); this property is not available if the default location ID is in use.
- `displayName`: display name of the subaccount (a string); this property is not available if there is no specified display name.
- `description`: subaccount description (a string); this property is not available if there is no description.
- `tunnel`: object outlining the current state of the tunnel.

Back to [Operations \[page 533\]](#)

Get Subaccount Configuration

URI

`/api/v1/configuration/subaccounts/
<regionHost>/<subaccount>`

Method

GET

Request

Response

```
{regionHost, subaccount,  
locationID, displayName,  
description, tunnel:{state,  
connections, applicationConnections:  
[], serviceChannels:[]}}
```

Errors

Roles

Administrator, Subaccount Administrator, Display, Support

Response Properties:

- `regionHost`: region host name (a string).

- `subaccount`: subaccount technical name (a string).
- `locationID`: location ID (a string); this property is not available if the default location ID is in use.
- `displayName`: display name of the subaccount (a string); this property is not available if there is no specified display name.
- `description`: description (a string); this property is not available if there is no description.
- `tunnel`: array of connection tunnels used by the subaccount.
 - `state`: *Connected*, *ConnectFailure*, or *Disconnected*
 - `connectedSinceTimeStamp`: connection start time as UTC timestamp
 - `connections`: number of subaccount connections
 - `applicationConnections`: array of connections to application instances
 - `serviceChannels`: type and state of the service channels used (types: HANA database, Virtual Machine or RFC)
 - `subaccountCertificate`: information on the subaccount certificate such as validity period, issuer and subject DN

Back to [Operations \[page 533\]](#)

Create Recovery Subaccount (Master Only)

⚠ Caution

Disaster recovery discontinued as of version 2.16. A `POST` request to `/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery` will trigger a 410 response.

Back to [Operations \[page 533\]](#)

Delete Recovery Subaccount (Master Only)

⚠ Caution

Disaster recovery discontinued as of version 2.16. A `DELETE` request to `/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery` will trigger a 410 response.

Back to [Operations \[page 533\]](#)

Refresh Certificate of Recovery Subaccount (Master Only)

⚠ Caution

Disaster recovery discontinued as of version 2.16. A `POST` request to `/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/validity` will trigger a 410 response.

Back to [Operations \[page 533\]](#)

Activate/Deactivate Recovery Subaccount (Master Only)

⚠ Caution

Disaster recovery discontinued as of version 2.16. A `PUT` request to `/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/state` will trigger a 410 response.

Back to [Operations \[page 533\]](#)

Takeover (Master Only)

⚠ Caution

Disaster recovery discontinued as of version 2.16. A `POST` request to `/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/takeover` will trigger a 410 response.

Back to [Operations \[page 533\]](#)

Synchronize Trust List (Master Only)

📌 Note

Available as of version 2.14.0.

URI	<code>/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/trust</code>
-----	--

Method	<code>POST</code>
--------	-------------------

Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

[Back to Operations \[page 533\]](#)

Get IDP Trust List

Note

Available as of version 2.14.0.

URI	<code>/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/trust/idps</code>
Method	<code>GET</code>
Request	
Response	<code>[{id, name, description, certificate, enabled}]</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each of which represents a trusted IDP through the following properties:

- `id`: (unique) ID of the trusted IDP (a number).
- `name`: name of the trusted IDP (a string).
- `description`: description of the trusted IDP (a string)
- `certificate`: object with the following properties: `issuer` (a string), `subjectDN` (a string), `notBeforeTimeStamp` (a UTC long number), and `notAfterTimeStamp` (a UTC long number).
- `enabled`: flag that indicates whether the IDP is enabled or disabled (that is, whether the IDP is trusted or not).

[Back to Operations \[page 533\]](#)

Get Application Trust List

Note

Available as of version 2.14.0.

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/trust/apps</code>
Method	<code>GET</code>
Request	
Response	<code>[{id, name, applicationType, enabled}]</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each of which represents a trusted application through the following properties:

- `id`: (unique) ID of the trusted application (a number).
- `name`: name of the trusted application (a string).
- `applicationType`: type of the application (a string, for example 'JAVA' or 'HANA').
- `enabled`: flag that indicates whether the application is enabled or disabled (that is, whether the application is trusted or not).

Back to [Operations \[page 533\]](#)

Enable Or Disable Trust (Master Only)

Replace `<type>` with the type (either `apps` or `idps`) that belongs to the specified `<id>`.

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/trust/<type>/ <id></code>
Method	<code>PUT</code>
Request	

Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Errors:

- NOT_FOUND (404): specified <id> does not exist for the given <type>.

Back to [Operations \[page 533\]](#)

Get IDP Trust Properties

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/trust/idps/ <id></code>
Method	<i>GET</i>
Request	
Response	<code>{id, name, description, certificate, enabled}</code>
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Response Properties:

- `id`: (unique) ID of the trusted IDP (a number).
- `name`: name of the trusted IDP (a string).
- `description`: description of the trusted IDP (a string).
- `certificate`: object with the following properties: `issuer` (a string), `subjectDN` (a string), `notBeforeTimeStamp` (a UTC long number), and `notAfterTimeStamp` (a UTC long number).
- `enabled`: flag that indicates whether the IDP is enabled or disabled (that is, whether the IDP is trusted or not).

Errors:

- NOT_FOUND (404): There is no trusted IDP with the given <id>

Back to [Operations \[page 533\]](#)

Get Application Trust Properties

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/trust/apps/ <id></code>
Method	<code>GET</code>
Request	
Response	<code>{id, name, applicationType, enabled}</code>
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Response Properties:

- `id`: (unique) ID of the trusted application (a number).
- `name`: name of the trusted application (a string).
- `applicationType`: type of the application (a string, for example 'JAVA' or 'HANA').
- `enabled`: flag that indicates whether the application is enabled or disabled (that is, whether the application is trusted or not).

Errors:

- NOT_FOUND (404): There is no trusted application with the given `<id>`.

Back to [Operations \[page 533\]](#)

1.2.2.5.9 System Mappings

Manage the Cloud Connector's system mappings via API.

Get All System Mappings

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMappings</code>
Method	<code>GET</code>

Request	
Response	<pre>[{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader, description, ...}]</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects with the following properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)
- `localHost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses (a string)
- `backendType`: Type of the backend (a string)
- `authenticationMode`: Authentication mode used on the backend side (a string).
- `hostInHeader`: Policy for setting the host in the response header. Available for HTTP(S) protocols only.
- `totalResourcesCount`: the total number of resources
- `enabledResourcesCount`: the number of enabled resources
- `description`: Description for the system mapping (a string).
- `sncPartnerName`: SNC name of an ABAP Server, required for RFCS communication only.
- `sapRouter`: SAP router route, required only if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedUsers`: Array of `{client, user}`, describing users that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

Get System Mapping

URI	<pre>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/ <virtualHost>:<virtualPort></pre>
-----	--

Method	GET
--------	---------------------

Request	
Response	<pre>{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader, description, ...}</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects with the following properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)
- `localHost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses (a string)
- `backendType`: Type of the backend (a string)
- `authenticationMode`: Authentication mode used on the backend side (a string).
- `hostInHeader`: Policy for setting the host in the response header (a string). Available for HTTP(S) protocols only.
- `totalResourcesCount`: the total number of resources
- `enabledResourcesCount`: the number of enabled resources
- `description`: Description for the system mapping (a string).
- `sncPartnerName`: SNC name of an ABAP Server, required for RFCS communication only.
- `sapRouter`: SAP router route, required only if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedUsers`: Array of `{client, user}`, describing users that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

Create System Mapping (Master Only)

URI	<code>/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings</code>
Method	<code>POST</code>

Request	<code>{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader, description, ...}</code>
Response	201 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request Properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)
- `localHost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses, which must be one of the following strings: `HTTP`, `HTTPS`, `RFC`, `RFCS`, `LDAP`, `LDAPS`, `TCP`, `TCPS`.
- `backendType`: Type of the backend system. Valid values are `abapSys`, `netweaverCE`, `netweaverGW`, `applServerJava`, `BC`, `PI`, `hana`, `otherSAPsys`, `nonSAPsys`.
- `authenticationMode`: Authentication mode to be used on the backend side, which must be one of the following strings: `NONE`, `NONE_RESTRICTED`, `X509_GENERAL`, `X509_RESTRICTED`, `KERBEROS`.
- `hostInHeader`: Policy for setting the host in the response header. This property is applicable to HTTP(S) protocols only, and it is **optional**. If set, it must be one of the following strings: `internal`, `virtual`. The default is `virtual`. You may also use all capital letters, i.e. `INTERNAL` and `VIRTUAL`.
- `description`: Description for the system mapping (string, optional). The default is no description, i.e. the empty string.
- `sncPartnerName`: SNC name of an ABAP Server, required for RFCS communication only.
- `sapRouter`: SAP router route, required only if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedUsers`: Array of `{client, user}`, describing users that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

Note

The authentication modes `NONE_RESTRICTED` and `X509_RESTRICTED` prevent the Cloud Connector from sending the system certificate in any case, whereas `NONE` and `X509_GENERAL` will send the system certificate if the circumstances allow it.

Delete System Mapping (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort></code>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Delete All System Mappings (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMappings</code>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Replace System Mapping (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort</code>
Method	<i>PUT</i>
Request	<pre>{virtualHost, virtualPort, localhost, localhost, protocol, backendType, authenticationMode, hostInHeader, description, ...}</pre>

Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request Properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)
- `localhost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses, which must be one of the following strings: `HTTP`, `HTTPS`, `RFC`, `RFCS`, `LDAP`, `LDAPS`, `TCP`, `TCPS`.
- `backendType`: Type of the backend system. Valid values are `abapSys`, `netweaverCE`, `netweaverGW`, `applServerJava`, `BC`, `PI`, `hana`, `otherSAPsys`, `nonSAPsys`.
- `authenticationMode`: Authentication mode to be used on the backend side, which must be one of the following strings: `NONE`, `NONE_RESTRICTED`, `X509_GENERAL`, `X509_RESTRICTED`, `KERBEROS`.
- `hostInHeader`: Policy for setting the host in the response header; this property is **optional**. If set, it must be one of the following strings: `internal`, `virtual`. The default is `virtual`. You may also use all capital letters, i.e. `INTERNAL` and `VIRTUAL`.
- `description`: Description for the system mapping (string, optional). The default is no description, i.e. the empty string.
- `sncPartnerName`: SNC name of an ABAP Server, only set for RFCS communication.
- `sapRouter`: SAP router route, only set if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedUsers`: Array of `{client, user}`, describing users, that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

📌 Note

The authentication modes `NONE_RESTRICTED` and `X509_RESTRICTED` prevent the Cloud Connector from sending the system certificate in any case, whereas `NONE` and `X509_GENERAL` will send the system certificate if the circumstances allow it.

1.2.2.5.10 System Mapping Resources

Manage the Cloud Connector's system mapping resources via API.

Operations

System Mapping Resources

- [Get all System Mapping Resources \[page 550\]](#)
- [Get System Mapping Resource \[page 551\]](#)
- [Create System Mapping Resource \[page 552\]](#)
- [Edit System Mapping Resource \[page 553\]](#)
- [Delete System Mapping Resource \[page 553\]](#)
- [Delete all System Mapping Resources \[page 554\]](#)

Allowed Clients

- [Get Allowed Clients for RFC System Mapping \[page 554\]](#)
- [Set Allowed Clients for RFC System Mapping \[page 555\]](#)
- [Add Allowed Clients for RFC System Mapping \[page 556\]](#)
- [Delete an Allowed Client for RFC System Mapping \[page 557\]](#)
- [Delete all Allowed Clients for RFC System Mapping \[page 557\]](#)

Blocked Users

- [Get Blocked Users for RFC System Mapping \[page 558\]](#)
 - [Set Blocked Users for RFC System Mapping \[page 559\]](#)
 - [Add Blocked Users for RFC System Mapping \[page 559\]](#)
 - [Remove one Blocked User for RFC System Mapping \[page 560\]](#)
 - [Remove all Blocked Users for RFC System Mapping \[page 561\]](#)
-

Get all System Mapping Resources

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort/ resources</code>
-----	--

Method	<code>GET</code>
--------	------------------

Request	
---------	--

Response	<pre>[{id, enabled, exactMatchOnly, websocketUpgradeAllowed, description}]</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each representing a resource through the following properties:

- `id`: The resource itself, which, depending on the owning system mapping, is either a URL path (or the leading section of it), or a RFC function name (prefix)
- `enabled`: Boolean flag indicating whether the resource is enabled.
- `exactMatchOnly`: Boolean flag determining whether access is granted only if the requested resource is an exact match.
- `websocketUpgradeAllowed`: Boolean flag indicating whether websocket upgrade is allowed; this property is of relevance only if the owning system mapping employs protocol HTTP or HTTPS.
- `description`: Description (a string); this property is not available unless explicitly set.

Back to [Top \[page 549\]](#)

Get System Mapping Resource

URI	<pre>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort/ resources/<encodedResourceId></pre>
Method	<i>GET</i>
Request	
Response	<pre>{id, enabled, exactMatchOnly, websocketUpgradeAllowed, description}</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response Properties:

- `id`: The resource itself, which, depending on the owning system mapping, is either a URL path (or the leading section of it), or a RFC function name (prefix)
- `enabled`: Boolean flag indicating whether the resource is enabled.

- `exactMatchOnly`: Boolean flag determining whether access is granted only if the requested resource is an exact match.
- `websocketUpgradeAllowed`: Boolean flag indicating whether websocket upgrade is allowed; this property is of relevance only if the owning system mapping employs protocol HTTP or HTTPS.
- `description`: Description (a string); this property is not available unless explicitly set.

Back to [Top \[page 549\]](#)

Create System Mapping Resource

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort/ resources</code>
Method	<code>POST</code>
Request	<code>{id, enabled, exactMatchOnly, websocketUpgradeAllowed, description}</code>
Response	201 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request Properties:

- `id`: The resource itself, which, depending on the owning system mapping, is either a URL path (or the leading section of it), or a RFC function name (prefix).
- `enabled`: Boolean flag indicating whether the resource is enabled (**optional**). The default value is `false`.
- `exactMatchOnly`: Boolean flag determining whether access is granted only if the requested resource is an exact match (**optional**). The default value is `false`.
- `websocketUpgradeAllowed`: Boolean flag indicating whether websocket upgrade is allowed (**optional**). The default value is `false`. This property is recognized only if the owning system mapping employs protocol HTTP or HTTPS.
- `description`: Description (a string, **optional**)

→ Tip

Encoded Resource ID

URI paths may contain the resource ID in order to identify the resource to be edited or deleted. A resource ID, however, may contain characters such as the forward slash that collide with the path separator of the

URI and hence require an escape mechanism. We adopted the following simple escape or encoding method for a resource ID:

1. Replace all occurrences of character '+' with '+2B'.
2. Replace all occurrences of character '-' with '+2D'.
3. Replace all occurrences of character '/' with '%2F'.

Back to [Top \[page 549\]](#)

Replace System Mapping Resource

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort/ resources/<encodedResourceId></code>
Method	<i>PUT</i>
Request	<pre>{enabled, exactMatchOnly, websocketUpgradeAllowed, description}</pre>
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request Properties:

- `enabled`: Boolean flag indicating whether the resource is enabled.
- `exactMatchOnly`: Boolean flag determining whether access is granted only if the requested resource is an exact match.
- `websocketUpgradeAllowed`: Boolean flag indicating whether websocket upgrade is allowed; this property is of relevance only if the owning system mapping employs protocol HTTP or HTTPS.
- `description`: Description (a string); this property is not available unless explicitly set.

Back to [Top \[page 549\]](#)

Delete System Mapping Resource

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort/ resources/<encodedResourceId>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Errors:

NOT_FOUND (404): Resource was not found

Back to [Top \[page 549\]](#)

Delete all System Mapping Resources

URI	/api/v1/configuration/subaccounts/ <region>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort>/resources
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Back to [Top \[page 549\]](#)

Get Allowed Clients for RFC System Mapping

Note

Available as of version 2.1.4.

Returns the list of allowed ABAP clients for the system mapping definition.

URI	<code>/api/v1/configuration/subaccounts/ <region>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort>/ allowedClients</code>
Method	<code>GET</code>
Request	
Response	<code>[client, ...]</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response Properties:

- Array of allowed clients:
 - `client`: ABAP client

Note

An empty list means that every client is allowed.

Back to [Top \[page 549\]](#)

Set Allowed Clients for RFC System Mapping

Note

Available as of version 2.1.4.

URI	<code>/api/v1/configuration/subaccounts/ <region>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort>/ allowedClients</code>
-----	---

Method	<i>POST</i>
Request	[client, ...]
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Request Properties:

- Array of allowed clients:
 - `client`: ABAP client

The existing list of available clients will be cleaned and populated by the provided in request.

Back to [Top \[page 549\]](#)

Add Allowed Clients for RFC System Mapping

Note

Available as of version 2.1.4.

Adds more ABAP clients to the allowed list for the system mapping definition.

URI	/api/v1/configuration/subaccounts/ <region>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort>/ allowedClients
Method	<i>PATCH</i>
Request	[client, ...]
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Request Properties:

- Array of allowed clients:
 - `client`: ABAP client

Clients provided in this request will be added to the existing list. Clients that are already listed will be ignored.

Back to [Top \[page 549\]](#)

Delete an Allowed Client for RFC System Mapping

Note

Available as of version 2.1.4.

URI	<code>/api/v1/configuration/subaccounts/ <region>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort>/ allowedClients/<client></code>
Method	<i>DELETE</i>
Request	
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Back to [Top \[page 549\]](#)

Delete all Allowed Clients for RFC System Mapping

Note

Available as of version 2.1.4.

URI	<code>/api/v1/configuration/subaccounts/ <region>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort>/ allowedClients</code>
Method	<i>DELETE</i>
Request	
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

ⓘ Note

An empty list means that every client is allowed.

Back to [Top \[page 549\]](#)

Get Blocked Users for RFC System Mapping

ⓘ Note

Available as of version 2.1.4.

Returns the list of blocked ABAP users for the system mapping definition.

URI	<code>/api/v1/configuration/subaccounts/ <region>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort>/ blockedClientUsers</code>
Method	<i>GET</i>
Request	
Response	<code>[{client, user}, ...]</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response Properties:

- Array of blocked users:
 - `client`: ABAP client
 - `user`: User name

Back to [Top \[page 549\]](#)

Set Blocked Users for RFC System Mapping

Note

Available as of version 2.1.4.

Sets or replaces the list of blocked ABAP users for the system mapping definition.

URI	<code>/api/v1/configuration/subaccounts/ <region>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort>/ blockedClientUsers</code>
Method	<i>POST</i>
Request	<code>[{client, user}, ...]</code>
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Request Properties:

- Array of blocked users:
 - `client`: ABAP client
 - `user`: User name

Back to [Top \[page 549\]](#)

Add Blocked Users for RFC System Mapping

Note

Available as of version 2.1.4.

Adds the provided list of blocked ABAP users to the current list.

URI	<code>/api/v1/configuration/subaccounts/ <region>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort>/ blockedClientUsers</code>
Method	<code>PATCH</code>
Request	<code>[{client, user}, ...]</code>
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Request Properties:

- Array of blocked users:
 - `client`: ABAP client
 - `user`: User name

Back to [Top \[page 549\]](#)

Remove one Blocked User for RFC System Mapping

Note

Available as of version 2.1.4.

Removes one user from the list of blocked ABAP users for the system mapping definition.

URI	<code>/api/v1/configuration/subaccounts/ <region>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort>/ blockedClientUsers/<client>:<user></code>
-----	---

Method	<i>DELETE</i>
Request	
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Back to [Top \[page 549\]](#)

Remove all Blocked Users for RFC System Mapping

Note

Available as of version 2.1.4.

Removes the list of blocked ABAP users for the system mapping definition.

URI	<code>/api/v1/configuration/subaccounts/ <region>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort>/ blockedClientUsers</code>
Method	<i>DELETE</i>
Request	
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Back to [Top \[page 549\]](#)

1.2.2.5.11 Domain Mappings

Manage the Cloud Connector's configuration for domain mappings via API.

Get Domain Mappings

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/domainMappings
Method	GET
Request	
Response	[{virtualDomain, internalDomain}]
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each representing a domain mapping through the following properties:

- `virtualDomain`: Domain used on the cloud side
- `internalDomain`: Domain used on the on-premise side

Create Domain Mapping (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/domainMappings
Method	POST
Request	{virtualDomain, internalDomain}
Response	201
Errors	
Roles	Administrator, Subaccount Administrator

Replace Domain Mapping (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ domainMappings/<internalDomain></code>
Method	<i>PUT</i>
Request	<code>{virtualDomain, internalDomain}</code>
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Request:

- `virtualDomain`: New virtual domain
- `internalDomain`: New internal domain

Errors:

- NOT_FOUND (404): Domain mapping does not exist.

Note

The internal domain in the URI path (i.e., `<internalDomain>`) is the current internal domain of the domain mapping that is to be edited. It may differ from the new internal domain set in the request.

Delete Domain Mapping (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ domainMappings/<internalDomain></code>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Errors:

- NOT_FOUND (404): Domain mapping does not exist.

Delete All Domain Mappings (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/domainMappings</code>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

1.2.2.5.12 Subaccount Service Channels

Manage Cloud Connector service channels via API.

Service Channel for HANA Database

- [Get all HANA Service Channels \[page 565\]](#)
- [Get Available HANA Channels \[page 566\]](#)
- [Get HANA Service Channel \[page 567\]](#)
- [Create HANA Service Channel \(Master Only\) \[page 568\]](#)
- [Replace HANA Service Channel \(Master Only\) \[page 568\]](#)

Service Channel for Virtual Machine

- [Get all Service Channels for Virtual Machines \[page 569\]](#)
- [Get Available Channels for Virtual Machines \[page 570\]](#)
- [Get Service Channel for a Virtual Machine \[page 570\]](#)
- [Create a Service Channel for a Virtual Machine \(Master Only\) \[page 571\]](#)
- [Replace a Service Channel for a Virtual Machine \(Master Only\) \[page 572\]](#)

Service Channel for ABAP Cloud

[Get all ABAP Cloud Service Channels \[page 573\]](#)

[Get ABAP Cloud Service Channel \[page 574\]](#)

[Create ABAP Cloud Service Channel \(Master Only\) \[page 575\]](#)

[Replace ABAP Cloud Service Channel \(Master Only\) \[page 575\]](#)

Service Channel for Kubernetes

Note

Available as of version 2.15.0.

[Get All Kubernetes Cluster Service Channels \[page 576\]](#)

[Get Kubernetes Service Channel \[page 577\]](#)

[Create Kubernetes Service Channel \(Master Only\) \[page 578\]](#)

[Replace Kubernetes Service Channel \(Master Only\) \[page 579\]](#)

General

[Enable/Disable Service Channel \(Master Only\) \[page 579\]](#)

[Delete Service Channel \(Master Only\) \[page 580\]](#)

[Delete all Service Channels \(Master Only\) \[page 581\]](#)

Deprecated

[Get all Service Channels \[page 581\]](#)

[Get Service Channel \[page 582\]](#)

[Create Service Channel \[page 583\]](#)

[Replace Service Channel \[page 584\]](#)

Get all HANA Service Channels

URI `/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/HANA`

Method `GET`

Request

Response

```
[{id, hanaInstanceName, instanceNumber, type, port, enabled, connections, state}]
```

Errors

Roles Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each of which represents a HANA service channel through the following properties:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1). This identifier is unique across all types of service channels.
- `hanaInstanceName`: name of the HANA instance (a string).
- `instanceNumber`: instance number.
- `type`: string 'HANA'.
- `port`: port of the HANA service channel (a number).
- `enabled`: boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.
- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties:
 - `connected` (a boolean flag indicating whether the channel is connected),
 - `openedConnections` (the number of open, possibly idle connections), and
 - `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).

Back to [Top \[page 564\]](#)

Get Available HANA Channels

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ availableChannels/HANA</code>
Method	<code>GET</code>
Request	
Response	<code>[{hanaInstanceName, version, type}]</code>
Errors	<code>RUNTIME_FAILURE</code>
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- `hanaInstanceName`: name of the HANA instance.
- `version`: version information.
- `type`: specific HANA database type (*not* the service channel type).

Errors:

- `RUNTIME_FAILURE` (500): the list of available HANA database instances could not be retrieved.

Back to [Top \[page 564\]](#)

Get HANA Service Channel

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/HANA/ <id></code>
Method	<code>GET</code>
Request	
Response	<pre>{id, hanaInstanceName, instanceNumber, type, port, enabled, connections, state}</pre>
Errors	<code>NOT_FOUND</code>
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1). This identifier is unique across all types of service channels.
- `hanaInstanceName`: name of the HANA instance (a string).
- `instanceNumber`: instance number.
- `type`: string 'HANA'.
- `port`: port of the HANA service channel (a number).
- `enabled`: boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.
- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties:
 - `connected` (a boolean flag indicating whether the channel is connected),
 - `openedConnections` (the number of open, possibly idle connections), and
 - `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).

Errors:

- `NOT_FOUND` (404): the HANA service channel with the given ID (that is, `<id>`) or the specified subaccount does not exist.

[Back to Top \[page 564\]](#)

Create HANA Service Channel (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/HANA</code>
Method	<code>POST</code>
Request	<pre>{hanaInstanceName, instanceNumber, connections}</pre>
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator, Subaccount Administrator

Request:

- `hanaInstanceName`: name of the HANA instance (a string).
- `instanceNumber`: instance number.
- `connections`: maximal number of open connections.

Errors:

- INVALID_REQUEST (400): invalid or out of range values.

[Back to Top \[page 564\]](#)

Replace HANA Service Channel (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/HANA/ <id></code>
Method	<code>PUT</code>
Request	<pre>{hanaInstanceName, instanceNumber, connections}</pre>

Response	204 on success
Errors	INVALID_REQUEST, NOT_FOUND
Roles	Administrator, Subaccount Administrator

Request:

- `hanaInstanceName`: name of the HANA instance (a string).
- `instanceNumber`: instance number.
- `connections`: maximal number of open connections.

Errors:

- INVALID_REQUEST (400): invalid or out of range values.
- NOT_FOUND (404): the HANA service channel with the given ID (that is, `<id>`) or the specified subaccount does not exist.

Back to [Top \[page 564\]](#)

Get all Service Channels for Virtual Machines

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ VirtualMachine</code>
Method	<code>GET</code>
Request	
Response	<code>[{id, name, endpointId, type, port, enabled, connections, state}]</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1). This identifier is unique across all types of service channels.
- `name`: name of the virtual machine on the cloud platform (a string).
- `endpointId`: unique ID (GUID) of the virtual machine (a string).
- `type`: string 'VirtualMachine'.
- `port`: port of the service channel for the virtual machine (a number).

- `enabled`: boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.
- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties:
 - `connected` (a boolean flag indicating whether the channel is connected),
 - `openedConnections` (the number of open, possibly idle connections), and
 - `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).

Back to [Top \[page 564\]](#)

Get Available Channels for Virtual Machines

URI	<code>/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/availableChannels/VirtualMachine</code>
Method	<code>GET</code>
Request	
Response	<code>[{endpointId, name}]</code>
Errors	<code>RUNTIME_FAILURE</code>
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- `name`: name of the virtual machine on the cloud platform (a string).
- `endpointId`: unique ID (GUID) of the virtual machine (a string).

Errors:

- `RUNTIME_FAILURE` (500): the list of available virtual machines could not be retrieved.

Back to [Top \[page 564\]](#)

Get Service Channel for a Virtual Machine

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ VirtualMachine/<id>
Method	GET
Request	
Response	{id, name, endpointId, type, port, enabled, connections, state}
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1). This identifier is unique across all types of service channels.
- `name`: name of the virtual machine on the cloud platform (a string).
- `endpointId`: unique ID (GUID) of the virtual machine (a string).
- `type`: string 'VirtualMachine'.
- `port`: port of the service channel for the virtual machine (a number).
- `enabled`: boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.
- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties:
 - `connected` (a boolean flag indicating whether the channel is connected),
 - `openedConnections` (the number of open, possibly idle connections), and
 - `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).

Errors:

- NOT_FOUND (404): the service channel for a virtual machine with the given ID (that is, <id>) or the specified subaccount does not exist.

Back to [Top \[page 564\]](#)

Create a Service Channel for a Virtual Machine (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ VirtualMachine
Method	POST
Request	{name, endpointId, port, connections}
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator, Subaccount Administrator

Request:

- `name`: name of the virtual machine on the cloud platform (a string).
- `endpointId`: unique ID (GUID) of the virtual machine (a string).
- `port`: port of the service channel for the virtual machine (a number).
- `connections`: maximal number of open connections.

Errors:

- INVALID_REQUEST (400): invalid or out of range values.

Back to [Top \[page 564\]](#)

Replace Service Channel for a Virtual Machine (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ VirtualMachine/<id>
Method	PUT
Request	{name, endpointId, connections}
Response	204 on success
Errors	INVALID_REQUEST, NOT_FOUND
Roles	Administrator, Subaccount Administrator

Request:

- `name`: name of the virtual machine on the cloud platform (a string).
- `endpointId`: unique ID (GUID) of the virtual machine (a string).
- `port`: port of the service channel for the virtual machine (a number).
- `connections`: maximal number of open connections.

Errors:

- `INVALID_REQUEST` (400): invalid or out of range values.
- `NOT_FOUND` (404): the service channel for a virtual machine with the given ID (that is, `<id>`) or the specified subaccount does not exist.

Back to [Top \[page 564\]](#)

Get all ABAP Cloud Service Channels

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ ABAPCloud</code>
Method	<code>GET</code>
Request	
Response	<pre>[{id, abapCloudTenantHost, instanceNumber, type, port, enabled, connections, state}]</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each of which represents an ABAP Cloud service channel through the following properties:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1). This identifier is unique across all types of service channels.
- `abapCloudTenantHost`: the host name (a string).
- `instanceNumber`: instance number.
- `type`: string 'ABAPCloud'.
- `port`: port of the ABAPCloud service channel (a number).
- `enabled`: boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.

- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties:
 - `connected` (a boolean flag indicating whether the channel is connected),
 - `openedConnections` (the number of open, possibly idle connections), and
 - `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).

Back to [Top \[page 564\]](#)

Get ABAP Cloud Service Channel

URI	<code>/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/ABAPCloud/<id></code>
Method	<code>GET</code>
Request	
Response	<code>{id, abapCloudTenantHost, instanceNumber, type, port, enabled, connections, state}</code>
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1). This identifier is unique across all types of service channels.
- `abapCloudTenantHost`: the host name (a string).
- `instanceNumber`: instance number.
- `type`: string 'ABAPCloud'.
- `port`: port of the ABAPCloud service channel (a number).
- `enabled`: boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.
- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties:
 - `connected` (a boolean flag indicating whether the channel is connected),
 - `openedConnections` (the number of open, possibly idle connections), and
 - `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).

Errors:

- NOT_FOUND (404): the ABAP Cloud service channel with the given ID (that is, <id>) or the specified subaccount does not exist.

Back to [Top \[page 564\]](#)

Create ABAP Cloud Service Channel (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ ABAPCloud</code>
Method	<i>POST</i>
Request	<code>{abapCloudTenantHost, instanceNumber, connections}</code>
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator, Subaccount Administrator

Request:

- `abapCloudTenantHost`: host name (a string).
- `instanceNumber`: instance number.
- `connections`: maximal number of open connections.

Errors:

- INVALID_REQUEST (400): invalid or out of range values.

Back to [Top \[page 564\]](#)

Replace ABAP Cloud Service Channel (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ ABAPCloud/<id></code>
-----	--

Method	<i>PUT</i>
Request	<code>{abapCloudTenantHost, instanceNumber, connections}</code>
Response	204 on success
Errors	INVALID_REQUEST, NOT_FOUND
Roles	Administrator, Subaccount Administrator

Request:

- `abapCloudTenantHost`: host name (a string).
- `instanceNumber`: instance number.
- `connections`: maximal number of open connections.

Errors:

- INVALID_REQUEST (400): invalid or out of range values.
- NOT_FOUND (404): the ABAP Cloud service channel with the given ID (that is, `<id>`) or the specified subaccount does not exist.

Back to [Top \[page 564\]](#)

Get All Kubernetes Cluster Service Channels

URI	<code>/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/K8S</code>
Method	<i>GET</i>
Request	
Response	<code>[{id, type, port, k8sCluster, k8sService, enabled, connections, state, comment}]</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each of which represents a service channel for a virtual machine through the following properties:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1); this identifier is unique across all types of service channels.
- `k8sCluster`: host name to access the Kubernetes cluster (a string).
- `k8sService`: host name providing the service inside of Kubernetes cluster (a string).
- `type`: the string 'K8S'.
- `port`: port of the service channel for the virtual machine (a number).
- `enabled`: Boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.
- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties `connected` (a Boolean flag indicating whether the channel is connected), `openedConnections` (the number of open, possibly idle connections), and `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).
- `comment`: comment or short description; this property is not supplied if no comment was provided.

Back to [Top \[page 564\]](#)

Get Kubernetes Service Channel

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/K8S/ <id></code>
Method	<code>GET</code>
Request	
Response	<pre>[{id, type, port, k8sCluster, k8sService, enabled, connections, state, comment}]</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1); this identifier is unique across all types of service channels.
- `k8sCluster`: host name to access the Kubernetes cluster (a string).

- `k8sService`: host name providing the service inside of Kubernetes cluster (a string).
- `type`: the string 'K8S'.
- `port`: port of the service channel for the virtual machine (a number).
- `enabled`: Boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.
- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties `connected` (a Boolean flag indicating whether the channel is connected), `openedConnections` (the number of open, possibly idle connections), and `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).
- `comment`: comment or short description; this property is not supplied if no comment was provided.

Back to [Top \[page 564\]](#)

Create Kubernetes Service Channel (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/K8S</code>
Method	<i>POST</i>
Request	<code>{k8sCluster, k8sService, port, connections, comment}</code>
Response	201
Errors	INVALID_REQUEST
Roles	Administrator, Subaccount Administrator

Request:

- `k8sCluster`: Kubernetes cluster host name, optionally with port separated by colon (a string).
- `k8sService`: service host inside the Kubernetes cluster (a string).
- `port`: local port.
- `connections`: maximal number of open connections.
- `comment`: optional comment or short description (a string).

Errors:

- INVALID_REQUEST (400): invalid or out-of-range values.

Back to [Top \[page 564\]](#)

Replace Kubernetes Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/K8S/ <id>
Method	PUT
Request	{k8sCluster, k8sService, port, connections, comment}
Response	INVALID_REQUEST, NOT_FOUND
Errors	
Roles	Administrator, Subaccount Administrator

Request:

- `k8sCluster`: Kubernetes cluster host name, optionally with port separated by colon (a string).
- `k8sService`: service host inside the Kubernetes cluster (a string).
- `port`: local port.
- `connections`: maximal number of open connections.
- `comment`: optional comment or short description (a string).

Errors:

- `INVALID_REQUEST` (400): invalid or out-of-range values.
- `NOT_FOUND` (404): the Kyma service channel with the given ID (that is, `<id>`) or the specified subaccount does not exist.

Back to [Top \[page 564\]](#)

Enable/Disable Service Channel (Master Only)

Use one of the following channel types to replace `<type>` in the URI: HANA, VirtualMachine, or ABAPCloud.

⚠ Caution

The URI variant without `<type>` still works, but it is obsolete and may be removed in a future release. We recommend that you move to using `<type>` as soon as possible.

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ <type>/<id>/state</code>
Method	<i>PUT</i>
Request	<code>{enabled}</code>
Response	
Errors	NOT_FOUND, RUNTIME_FAILURE
Roles	Administrator, Subaccount Administrator

Errors:

- NOT_FOUND (404): service channel (or subaccount) does not exist.
- RUNTIME_FAILURE (500): service channel could not be opened (when attempting to set `enabled:true`).

Back to [Top \[page 564\]](#)

Delete Service Channel (Master Only)

⚠ Caution

The URI variant without `<type>` still works, but it is obsolete and may be removed in a future release. We recommend to move to using `<type>` as soon as possible.

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ <type>/<id></code>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Errors:

- NOT_FOUND (404): service channel (or subaccount) does not exist.

Back to [Top \[page 564\]](#)

Delete all Service Channels (Master Only)

Note

Omit `<type>` to delete all service channels, regardless of the type. To delete all service channels of a particular type, replace `<type>` with one of the following channel types: HANA, VirtualMachine, or ABAPCloud.

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ <type>]</code>
Method	<code>DELETE</code>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Back to [Top \[page 564\]](#)

Get all Service Channels

Caution

Obsolete. This API is deprecated and may be removed in a future release. Use the getters for the specific service channel type (that is, HANA (database), Virtual Machine, or ABAP Cloud) which provide properties tailored for the respective channel type.

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels</code>
Method	<code>GET</code>
Request	

Response

```
[{typeDesc, details, port,
enabled, connected, connectionsCount,
availableConnectionsCount,
connectedSinceTimeStamp}]
```

Errors

Roles

Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each of which represents a service channel through the following properties:

- `typeDesc`: an object specifying the service channel type through the properties `typeKey` and `typeName`.
- `details`
- `port`
- `enabled`
- `connected`
- `connectionsCount`
- `availableConnectionsCount`
- `connectedSinceTimeStamp`

Back to [Top \[page 564\]](#)

Get Service Channel

⚠ Caution

Obsolete. This API is deprecated and may be removed in a future release. Use the getters for the specific service channel type (that is, HANA (database), Virtual Machine, or ABAP Cloud) which provide properties tailored for the respective channel type.

URI

```
/api/v1/configuration/subaccounts/
<regionHost>/<subaccount>/channels/<id>
```

Method

GET

Request

Response

```
[{typeDesc, details, port,
enabled, connected, connectionsCount,
availableConnectionsCount,
connectedSinceTimeStamp}]
```

Errors

Roles

Administrator, Subaccount Administrator, Display, Support

Response:

- `typeDesc`: an object specifying the service channel type through the properties `typeKey` and `typeName`.
- `details`
- `port`
- `enabled`
- `connected`
- `connectionsCount`
- `availableConnectionsCount`
- `connectedSinceTimeStamp`

Back to [Top \[page 564\]](#)

Create Service Channel

⚠ Caution

Obsolete. This API is deprecated and may be removed in a future release. Create a service channel using the API for the respective channel type, that is, HANA (database), Virtual Machine, or ABAP Cloud.

URI	<code>/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels</code>
Method	<i>POST</i>
Request	<code>{typeKey,details,serviceNumber,connectionCount}</code>
Response	201 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request Properties:

- `typeKey`: type of service channel. Valid values are HANA_DB, HCPVM, RFC.
- `details`:
 - HANA instance name for HANA_DB
 - VM name for HCPVM

- S/4HANA Cloud tenant host for RFC
- `serviceNumber`: service number, which is mapped to a port according to the type of service channel.
- `connectionCount`: number of connections for the channel.

Back to [Top \[page 564\]](#)

Replace Service Channel

⚠ Caution

Obsolete. This API is deprecated and may be removed in a future release. Replace a service channel using the API for the respective channel type, that is, HANA (database), Virtual Machine, or ABAP Cloud.

URI	<code>/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/<id></code>
Method	<i>PUT</i>
Request	<code>{typeKey,details,serviceNumber,connectionCount}</code>
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Back to [Top \[page 564\]](#)

1.2.2.5.13 Access Control Entities

Manage RFC-specific access control entities for the Cloud Connector via API.

[Get List Of Allowed Clients \[page 585\]](#)

[Get List Of Blocked Client/User Pair \[page 585\]](#)

[Create Allowed Clients \(Master Only\) \[page 586\]](#)

[Extend Allowed Clients \(Master Only\) \[page 587\]](#)

[Create a Blocked Client/User Pair \(Master Only\) \[page 587\]](#)

[Extend a Blocked Client/User Pair \(Master Only\) \[page 588\]](#)

[Delete All Allowed Clients \(Master Only\) \[page 588\]](#)

[Delete All Blocked Client/User Pairs \(Master Only\) \[page 589\]](#)

[Remove an Item from the List of Allowed Clients \(Master Only\) \[page 589\]](#)

[Remove Items on the List of Blocked Client/User Pairs for a Given User \(Master Only\) \[page 590\]](#)

[Remove Items on the List of Blocked Client/User Pairs for a Given Client \(Master Only\) \[page 590\]](#)

[Remove an Item on the List of Blocked Client/User Pairs \(Master Only\) \[page 591\]](#)

Get List Of Allowed Clients

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ allowedClients</code>
Method	<code>GET</code>
Request	
Response	<code>[{client}]</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

List of objects, each representing an allowed client:

- `client`: client name

[Back to Top \[page 584\]](#)

Get List Of Blocked Client/User Pair

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ blockedClientUsers</code>
-----	--

Method	GET
Request	
Response	<code>[{client, user}]</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

List of objects, each representing a blocked client and user:

- `client`: client name
`user`: user name

Back to [Top \[page 584\]](#)

Create Allowed Clients (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ allowedClients</code>
Method	POST
Request	<code>[client]</code>
Response	201 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request:

List of strings, each representing an allowed client.

Back to [Top \[page 584\]](#)

Extend Allowed Clients (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ allowedClients</code>
Method	<i>PATCH</i>
Request	<code>[client]</code>
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request:

List of strings, each representing an allowed client.

Back to [Top \[page 584\]](#)

Create a Blocked Client/User Pair (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ blockedClientUsers</code>
Method	<i>POST</i>
Request	<code>[{client, user}]</code>
Response	201 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request:

List of objects, each representing a blocked client and user:

- `client`: ABAP client
- `user`: ABAP user

Back to [Top \[page 584\]](#)

Extend a Blocked Client/User Pair (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ blockedClientUsers</code>
Method	<i>PATCH</i>
Request	<code>[{client, user}]</code>
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request:

List of objects, each representing a blocked client and user:

- `client`: ABAP client
- `user`: ABAP user

Back to [Top \[page 584\]](#)

Delete All Allowed Clients (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ allowedClients</code>
-----	--

Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Back to [Top \[page 584\]](#)

Delete All Blocked Client/User Pairs (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ blockedClientUsers</code>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Back to [Top \[page 584\]](#)

Remove an Item from the List of Allowed Clients (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ allowedClients/<client></code>
Method	<i>DELETE</i>

Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Back to [Top \[page 584\]](#)

Remove Items on the List of Blocked Client/User Pairs for a Given User (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ blockedClientUsers/user/<user></code>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Back to [Top \[page 584\]](#)

Remove Items on the List of Blocked Client/User Pairs for a Given Client (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ blockedClientUsers/client/<client></code>
Method	<i>DELETE</i>

Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Back to [Top \[page 584\]](#)

Remove an Item on the List of Blocked Client/User Pairs (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ blockedClientUsers/client/ <client>:<user></code>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Back to [Top \[page 584\]](#)

1.2.2.5.14 Examples

Find examples on how to use the Cloud Connector's configuration REST APIs.

Concept

The sample code in this section ([download](#) zip file) demonstrates how to use the REST APIs provided by Cloud Connector to perform various configuration tasks.

Starting with a freshly installed Cloud Connector, the samples include initial configuration of the Cloud Connector instance, connectivity setup, high availability configuration, and common tasks like backup/restore operations, as well as integration with solution management.

The examples are implemented in *Kotlin*, a simple Java VM-based language. However, even if you are using a different language, they still show the basic use of the APIs and their parameters for specific configuration purposes.

If you are not familiar with Kotlin, find a brief introduction and some typical statements below.

[REST API Parameters \[page 592\]](#)

[REST API Invocation \[page 593\]](#)

[How To Use the Examples \[page 593\]](#)

REST API Parameters

In almost all requests and responses, structures are encoded in JSON format. To describe the parameter details, we use Kotlin data classes.

This class represents a structure that you can use as value in a request or response:

```
data class OnlyPropertiesNamesAreRelevant(  
    val user: String,  
    val password: String  
)
```

The JSON representation for that class is

```
{"user":<userValue>, "password":<passwordValue>}
```

Encoded in Kotlin this string looks like

```
"""{"user":"$userValue", "password":"$passwordValue"}"""
```

or as an object:

```
val credentials = OnlyPropertiesNamesAreRelevant(userValue, passwordValue)
```

Back to [Concept \[page 591\]](#)

REST API Invocation

```
(a) Fuel.put(url)
(b)   .header("Connection", "close")
(c)   .authentication().basic(user, password)
(d)   .jsonBody(credentials)
(e)   .responseObject<OnlyPropertiesNamesAreRelevant> { _, _, result ->
(f)       when (result) {
           is Result.Failure -> processRequestError(result.error)
           is Result.Success -> println("returned: ${result.get()}")
         }
      }
      .join()
```

- (a) - *Fuel* is an HTTP framework used in the examples. Important is the verb after *Fuel* - it is the REST API method.
- (b) - Adds a request header `Connection: close`, which forces a connection close after request. In the examples, this header is defined on *FuelManager* for all calls.
- (c) - Basic authentication is used for the call with `user` and `password`.
- (d) - HTTP requests with parameters require mostly JSON. The `jsonBody`-method adds the header `Content-Type: application/json` and serializes the provided object `credentials` to JSON. Requests without body like *DELETE* or *GET* omit body methods.
- (e) - The `responseObject<ClassType>`-method adds the header `Accept: application/json` and de-serializes the response to the specified class type. Some APIs do not have any response or non-JSON response. In such cases, the method `response` is used instead of `responseObject<ClassType>`.
- (f) - The way, how Kotlin decides between success (2xx) and failed responses. For details on the possible response status, see [Configuration REST APIs \[page 482\]](#).

Back to [Concept \[page 591\]](#)

How To Use the Examples

Some common details, used by different examples, were extracted to the `scenario.json` configuration file. This lets you use meaningful names like `config.master!!.user` in the examples. The file is loaded by

```
val config = loadScenarioConfiguration()
```

Each example also invokes

```
disableTrustChecks()
```

This method disables all SSL-related checks.

⚠ Caution

`disableTrustChecks()` is only used for test purposes. *Do not* use it in a productive environment.

Both methods, as well as some common REST API parameter structures (data classes) are defined in the file [Scenario Configuration \[page 595\]](#). This is the only help class under `sources/`.

When using the examples, start with [Initial Configuration \[page 596\]](#).

Prerequisite is a freshly installed Cloud Connector.

After a mandatory password change and defining the high availability role of the Cloud Connector instance (master or shadow), the example demonstrates how to provide a description for the instance and how to set up the UI and system certificates.

Once the initial configuration is done, you can optionally proceed with these steps:

- Connect to your subaccount on BTP ([Subaccount Configuration \[page 598\]](#))
- Create or restore a configuration backup ([Backup And Restore Configuration \[page 605\]](#))
- Configure and connect high availability instances ([High Availability Settings \[page 603\]](#))
- Integrate the Cloud Connector with the solution management infrastructure ([Solution Management Integration \[page 606\]](#))

Back to [Concept \[page 591\]](#)

Related Information

[scenario.json \[page 594\]](#)

[Source Files \[page 595\]](#)

1.2.2.5.14.1 scenario.json

Sample Code

```
{
  "subaccount": {
    "regionHost": "cf.eu10.hana.ondemand.com",
    "subaccount": "11aabbcc-7821-448b-9ecf-a7d986effa7c",
    "user": "xxx",
    "password": "xxx"
  },
  "master": {
    "url": "https://localhost:8443",
    "user": "Administrator",
    "password": "test"
  },
  "shadow": {
    "url": "https://localhost:8444",
    "user": "Administrator",
    "password": "test"
  }
}
```

1.2.2.5.14.2 Source Files

[Scenario Configuration \[page 595\]](#)

[Initial Configuration \[page 596\]](#)

[Subaccount Configuration \[page 598\]](#)

[High Availability Settings \[page 603\]](#)

[Backup And Restore Configuration \[page 605\]](#)

[Solution Management Integration \[page 606\]](#)

1.2.2.5.14.2.1 Scenario Configuration

Sample Code

```
package com.sap.scc.examples
import com.github.kittinunf.fuel.core.FuelError
import com.google.gson.Gson
import java.io.File
import java.security.SecureRandom
import java.security.cert.X509Certificate
import javax.net.ssl.*
data class CloudConnector(
    val url: String,
    var user: String,
    var password: String
)
fun disableTrustChecks() {
    try {
        HttpURLConnection.setDefaultHostnameVerifier { hostname: String,
session: SSLSession -> true }
        val context: SSLContext = SSLContext.getInstance("TLS")
        val trustAll: X509TrustManager = object : X509TrustManager {
            override fun checkClientTrusted(chain: Array<X509Certificate>,
authType: String) {}
            override fun checkServerTrusted(chain: Array<X509Certificate>,
authType: String) {}
            override fun getAcceptedIssuers(): Array<X509Certificate> {
                return arrayOf()
            }
        }
        context.init(null, arrayOf(trustAll), SecureRandom())
        HttpURLConnection.setDefaultSSLSocketFactory(context.socketFactory)
    } catch (e: Exception) {
        e.printStackTrace()
    }
}
class ScenarioConfiguration {
    var subaccount: SubaccountParameters? = null
    var master: CloudConnector? = null
    var shadow: CloudConnector? = null
}
data class SubaccountParameters(
    val regionHost: String,
```

```

        val subaccount: String,
        val user: String,
        val password: String,
        val locationId: String? = null
    )
    data class SccCertificate(
        val subjectDN: String? = null,
        val issuer: String? = null,
        val notAfter: String? = null,
        val notBefore: String? = null,
        val subjectAltNames: List<SubjectAltName>? = null
    )
    data class SubjectAltName(
        val type: String? = null,
        val value: String? = null
    )
    internal fun loadScenarioConfiguration(): ScenarioConfiguration {
        println("scenario.json will be loaded from $
{File("scenario.json").absolutePath}")
        return Gson().fromJson(File("scenario.json").readText(),
ScenarioConfiguration::class.java)
    }
    internal fun processRequestError(error: FuelError) {
        println("failed with ${error.message} ${String(error.errorData)}")
        throw RuntimeException("Stop here. ")
    }
}

```

1.2.2.5.14.2.2 Initial Configuration

Sample Code

```

package com.sap.scc.examples
//import com.sap.scc.examples.SccCertificate
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.BlobDataPart
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.Method
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
import java.io.ByteArrayInputStream
import java.io.File
/*
 * This example shows how to use REST APIs to perform the initial configuration
 * of a master instance
 * after installing and starting the Cloud Connector.
 * As a prerequisite you need to install and start the Cloud Connector.
 * The example begins with changing the initial password, setting the instance
 * to the master role, edit the description,
 * and upload UI and system certificates.
 * For the certificates used by Cloud Connector in order to access the UI and
 * for the system certificate used to access backend systems,
 * we simply upload the already available PKCS#12 certificates uiCert.p12 and
 * systemCert.p12 encrypted with the password "test1234".
 * Cloud Connector also provides other options for certificate management,
 * please take a look at the documentation.
 * The configuration details for master and shadow instances can be found in
 * scenario.json.
 */

```

```

fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
    certificate.
    //So for this demonstration use case we need to deactivate all trust
    checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
    efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
    //Load configuration from property file
    val config = loadScenarioConfiguration()
    //Change the initial password
    Fuel.put("${config.master!!.url}/api/v1/configuration/connector/
    authentication/basic")
        .authentication().basic(config.master!!.user, "manage")
        .body("""{"oldPassword":"manage", "newPassword":"$
    {config.master!!.password}""")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("Password successfully set to $
    {config.master!!.password}")
            }
        }
        .join()
    //Set the High-Availability Role to master, use "shadow" if you want to
    set it to shadow
    Fuel.put("${config.master!!.url}/api/v1/configuration/connector/haRole")
        .authentication().basic(config.master!!.user,
    config.master!!.password)
        .body("master")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("high-availability role
    successfully set to 'master'")
            }
        }
        .join()
    //Edit Common Description
    Fuel.put("${config.master!!.url}/api/v1/configuration/connector")
        .authentication().basic(config.master!!.user,
    config.master!!.password)
        .body("""{"description":"<description of Cloud Connector
    instance>""")
        .responseObject<SccDescriptionResponse> { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println(result.get())
            }
        }
        .join()
    //Upload a PKCS#12 Certificate as UI Certificate
    val uiCertificateFormData = listOf("password" to "test1234",
    "keyPassword" to "test1234")
    Fuel.upload(
        "${config.master!!.url}/api/v1/configuration/connector/ui/
    uiCertificate",
        Method.PUT,
        uiCertificateFormData
    )
        .add(BlobDataPart(ByteArrayInputStream(File("uiCert.p12").readBytes())
    , name = "pkcs12"))
        .authentication().basic(config.master!!.user,
    config.master!!.password)
        .response { _, _, result ->

```

```

        when (result) {
            is Result.Failure -> processRequestError(result.error)
            is Result.Success -> println("PKCS#12 Certificate
'uiCert.p12' successfully uploaded")
        }
    }
    .join()
    //Upload a PKCS#12 Certificate as as System Certificate
    val systemCertificateFormData = listOf("password" to "test1234",
"keyPassword" to "test1234")
    Fuel.upload(
        "${config.master!!.url}/api/v1/configuration/connector/onPremise/
systemCertificate",
        Method.PUT,
        systemCertificateFormData
    )
    .add(BlobDataPart(ByteArrayInputStream(File("systemCert.p12").readByte
s()), name = "pkcs12"))
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .response { _, _, result ->
        when (result) {
            is Result.Failure -> processRequestError(result.error)
            is Result.Success -> println("PKCS#12 Certificate
'systemCert.p12' successfully uploaded")
        }
    }
    .join()
}
//Data structures used by REST calls in this scenario
data class SccDescriptionResponse(
    var role: String,
    var description: String
)

```

1.2.2.5.14.2.3 Subaccount Configuration

↔ Sample Code

```

package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.jsonBody
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
/*
This example shows how to use REST APIs to configure and connect a
subaccount in cloud connector.
The example begins with (1) connecting of the subaccount, then we create a
system (2) for an HTTP service
and (3) for an RFC service.

The configuration details for master and shadow instances can be found in
scenario.json.
*/
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
certificate.

```

```

//So for this demonstration use case we need to deactivate all trust
checks.
disableTrustChecks()
//Use 'Connection: close' header, to make stateless communication more
efficient
FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
//Add output of cURL commands for revision
//FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
//1.1. Create and connect subaccount
//Load configuration from property file
val config = loadScenarioConfiguration()
//Some cloud regions require 2-Factor-Authentication
println("Enter MFA (aka 2FA) token, if required: ")
var token = readLine() ?: ""
//Parameters required to establish the connection to the subaccount (aka
the secure tunnel)
var subaccountCreateData = SubaccountConfiguration(
    config.subaccount!!.regionHost, config.subaccount!!.subaccount,
    config.subaccount!!.user, "" + config.subaccount!!.password + token,
locationId = config.subaccount!!.locationId
)
//Optional: Initialize the map to work with generated _links
//If you don't like to use _links, you can easily compute the entity links
var subaccountLinks: Map<String, HalLink> = mapOf()
Fuel.post("${config.master!!.url}/api/v1/configuration/subaccounts")
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .jsonBody(subaccountCreateData)
    .responseObject<SubaccountInfo> { _, response, result ->
        when (result) {
            is Result.Success -> {
                val subaccountLocation =
response.header("Location").first()
                println("1.1. subaccount was created under:
$subaccountLocation")
                println("subaccount: ${result.get()} ")
                //GET details as SubaccountInfo every time possible by
invoking
                //https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>
                subaccountLinks = result.get()._links
            }
            is Result.Failure -> processRequestError(result.error)
        }
    }
    .join()
//1.2. From time to time it is necessary to extend the validity of the
subaccount - refresh subaccount
//Again some cloud landscapes require 2-Factor-Authentication
println("Enter MFA (aka 2FA) token for refresh, if required: ")
token = readLine() ?: ""
Fuel.post(subaccountLinks["validity"]!!.href)
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .jsonBody(SubaccountRefreshCred(config.subaccount!!.user, "" +
config.subaccount!!.password + token))
    .responseObject<SubaccountInfo> { _, _, result ->
        when (result) {
            is Result.Success -> println("1.2. validity of the subaccount
was extended")
            is Result.Failure -> processRequestError(result.error)
        }
    }
    .join()
//2.1. Create a system mapping (aka access control) for an HTTP service
val httpSystem = SystemMapping(
    "virtual.host", "vport", "local", "lport",
CommunicationProtocol.HTTPS,

```

```

        BackendType.abapSys, hostInHeader = HostInHeader.INTERNAL
    )
    var httpSystemLinks: Map<String, Hallink> = mapOf()
    Fuel.post(subaccountLinks["systemMappings"]!!.href)
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .jsonBody(httpSystem)
        .responseString { _, response, result ->
            when (result) {
                is Result.Success -> {
                    val httpSystemMappingLocation =
response.header("Location").first()
                    println("2.1. system mapping was created under:
$httpSystemMappingLocation")
                    //GET the details about the system mapping by invoking
//https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>
                    Fuel.get(httpSystemMappingLocation)
                        .authentication().basic(config.master!!.user,
config.master!!.password)
                        .responseObject<SystemMapping> { _, _, result ->
                            when (result) {
                                is Result.Success -> {
                                    println("system mapping: ${result.get()}")
                                    httpSystemLinks = result.get()._links
                                }
                                is Result.Failure ->
processRequestError(result.error)
                            }
                        }
                    }
                }
            }
            .join()
        }
        is Result.Failure -> processRequestError(result.error)
    }
    .join()
    //2.2 Add an allowed resource to the system mapping. Since this system
mapping points to an HTTP service, use HTTP resource parameters
    Fuel.post(httpSystemLinks["resources"]!!.href)
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .jsonBody(HttpResource("/", exactMatchOnly = false))
        .responseString { _, response, result ->
            when (result) {
                is Result.Success -> {
                    val httpResourceLocation =
response.header("Location").first()
                    println("2.2. http resource was created under:
$httpResourceLocation")
                    //GET the details about the resource by invoking
//https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>/<encoded-id>
//<encoded-id> -> replace '/' with '-'
                    Fuel.get(httpResourceLocation)
                        .authentication().basic(config.master!!.user,
config.master!!.password)
                        .responseObject<HttpResource> { _, _, result ->
                            when (result) {
                                is Result.Success -> println("http resource:
${result.get()}")
                                is Result.Failure ->
processRequestError(result.error)
                            }
                        }
                    }
                }
            }
            .join()
        }
        is Result.Failure -> processRequestError(result.error)
    }
}
}

```

```

    }
    .join()
    //3.1 Create a system mapping for an RFC service
    var rfcSystemLinks: Map<String, Hallink> = mapOf()
    Fuel.post(subaccountLinks["systemMappings"]!!.href)
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .jsonBody(
            SystemMapping(
                "virtual.host",
                "rfcport",
                "local",
                "rfcport",
                CommunicationProtocol.RFCS,
                BackendType.abapSys
            )
        )
        .responseString { _, response, result ->
            when (result) {
                is Result.Success -> {
                    val rfcSystemMappingLocation =
response.header("Location").first()
                    println("3.1. system mapping was created under:
$rfcSystemMappingLocation")
                    //GET the details about the system mapping by invoking
//https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>
                    Fuel.get(rfcSystemMappingLocation)
                        .authentication().basic(config.master!!.user,
config.master!!.password)
                        .responseObject<SystemMapping> { _, _, result ->
                            when (result) {
                                is Result.Success -> {
                                    println("system mapping: ${result.get()}")
                                    rfcSystemLinks = result.get()._links
                                }
                                is Result.Failure ->
processRequestError(result.error)
                            }
                        }
                    }
                }
            }
        }
        .join()
        }
        is Result.Failure -> processRequestError(result.error)
    }
}
.join()
//3.2 Now add the function module RFC_SYSTEM_INFO as an allowed resource
to the system mapping
val rfcResource = RfcResource("RFC_SYSTEM_INFO")
Fuel.post(rfcSystemLinks["resources"]!!.href)
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .jsonBody(rfcResource)
    .responseString { _, response, result ->
        when (result) {
            is Result.Success -> {
                val resourceLocation = response.header("Location").first()
                println("3.2. rfc resource was created under:
$resourceLocation")
                //GET the details about the resource by invoking
//https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>/<encoded-id>
                //<encoded-id> -> replace '/' with '-'
                Fuel.get(resourceLocation)
                    .authentication().basic(config.master!!.user,
config.master!!.password)
                    .responseObject<RfcResource> { _, _, result ->
                        when (result) {

```



```

        is Result.Success -> println("rfc resource: $
{result.get()}")
        is Result.Failure ->
processRequestError(result.error)
    }
    }
    .join()
}
    is Result.Failure -> processRequestError(result.error)
}
}
.join()
}
//Data structures used by REST calls in this scenario
//Nullable properties are optional
data class SubaccountConfiguration(
    var regionHost: String,
    var subaccount: String,
    var cloudUser: String,
    var cloudPassword: String,
    var displayName: String? = null,
    var locationId: String? = null
)
data class SubaccountInfo(
    val displayName: String,
    val regionHost: String,
    val subaccount: String,
    val tunnel: SubaccountTunnelInfo,
    val user: String,
    val _links: Map<String, HalLink>
)
data class SubaccountTunnelInfo(
    val state: String, //Connected
    val connectedSince: String, //timestamp formatted with client locale
    val connections: Int,
    val applicationConnections: List<String>,
    val serviceChannels: List<String>,
    val subaccountCertificate: X509CertificateInfo,
)
data class X509CertificateInfo(
    val notAfter: String,
    val notBefore: String,
    val subjectDN: String,
    val issuer: String
)
data class HalLink(val href: String)
data class SubaccountRefreshCred(
    var cloudUser: String,
    var cloudPassword: String
)
data class SystemMapping(
    val virtualHost: String,
    val virtualPort: String,
    val localHost: String,
    val localPort: String,
    val protocol: CommunicationProtocol,
    val backendType: BackendType,
    val sncPartnerName: String? = null,
    val hostInHeader: HostInHeader? = null,
    val sapRouter: String? = null,
    val authenticationMode: AuthenticationMode = AuthenticationMode.NONE,
    val description: String = "",
) {
    val _links: Map<String, HalLink> = mapOf()
}
enum class CommunicationProtocol {
    HTTP, HTTPS, RFC, RFCS, LDAP, LDAPS, TCP, TCPS
}

```

```

enum class BackendType {
    abapSys, netweaverCE, applServerJava, BC, PI, hana, netweaverGW,
    otherSAPsys, nonSAPsys
}
enum class HostInHeader {
    INTERNAL, VIRTUAL
}
enum class AuthenticationMode {
    NONE, X509_CERTIFICATE, X509_CERTIFICATE_LOCAL, KERBEROS
}
data class HttpResource(
    val id: String, //URI path
    val enabled: Boolean = true,
    val exactMatchOnly: Boolean = true,
    val websocketUpgradeAllowed: Boolean = true,
    val description: String = ""
)
data class RfcResource(
    val id: String, //Function module or prefix for function module
    val enabled: Boolean = true,
    val exactMatchOnly: Boolean = true,
    val description: String = ""
)

```

1.2.2.5.14.2.4 High Availability Settings

Sample Code

```

package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.Headers
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.jsonBody
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
import java.net.URL
/*
 * This example shows how to use REST APIs to change high availability settings
 * of the Cloud Connector instances.
 * As a prerequisite you need to install and start a shadow and a master
 * instance.
 * Afterwards perform the initial configuration of the master instance (see
 * InitialConfiguration.kt).
 *
 * The configuration details for master and shadow instances can be found in
 * scenario.json.
 */
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
    //certificate.
    //So for this demonstration use case we need to deactivate all trust
    //checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
    //efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
}

```

```

//Load configuration from property file
val config = loadScenarioConfiguration()
//The high-availability role 'master' in Cloud Connector instance
'master' is already set in the initial configuration.
//Set the high-availability role to 'shadow' in Cloud Connector instance
'shadow'
Fuel.put("${config.shadow!!.url}/api/v1/configuration/connector/haRole")
    .authentication().basic(config.shadow!!.user,
config.shadow!!.password)
    .body("shadow")
    .response { _, _, result ->
        when (result) {
            is Result.Failure -> processRequestError(result.error)
            is Result.Success -> println("high-availability role
successfully set to 'shadow'")
        }
    }
.join()
//Enable HA in the master instance and define the allowed shadow hosts
Fuel.put("${config.master!!.url}/api/v1/configuration/connector/ha/master/
config")
    .header(Headers.CONTENT_TYPE, "application/json")
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .body("""{"haEnabled":"true", "allowedShadowHost":"$
{URL(config.shadow!!.url).host}""")
    .responseObject<HAMasterConfiguration> { _, _, result ->
        when (result) {
            is Result.Failure -> processRequestError(result.error)
            is Result.Success -> {
                val masterConfig = result.get()
                println("Master configuration: $masterConfig ")
            }
        }
    }
.join()
//The configuration of the shadow instance
var shadowConfig: HShadowConfiguration? = null
//Get the current configuration
Fuel.get("${config.shadow!!.url}/api/v1/configuration/connector/ha/shadow/
config")
    .authentication().basic(config.shadow!!.user,
config.shadow!!.password)
    .responseObject<HShadowConfiguration> { _, _, result ->
        when (result) {
            is Result.Failure -> processRequestError(result.error)
            is Result.Success -> shadowConfig = result.get()
        }
    }
.join()
//Edit the retrieved configuration and set it
shadowConfig!!.masterPort = "${URL(config.master!!.url).port}"
shadowConfig!!.masterHost = URL(config.master!!.url).host
shadowConfig!!.ownHost = URL(config.master!!.url).host
Fuel.put("${config.shadow!!.url}/api/v1/configuration/connector/ha/shadow/
config")
    .authentication().basic(config.shadow!!.user,
config.shadow!!.password)
    .jsonBody(shadowConfig!!)
    .responseObject<HShadowConfiguration> { _, _, result ->
        when (result) {
            is Result.Failure -> processRequestError(result.error)
            is Result.Success -> shadowConfig = result.get()
        }
    }
.join()
//Set the HA link of both cloud connector instances to 'CONNECT'

```

```

    Fuel.post("${config.shadow!!.url}/api/v1/configuration/connector/ha/
shadow/state")
        .header(Headers.CONTENT_TYPE, "application/json")
        .authentication().basic(config.shadow!!.user,
config.shadow!!.password)
        .body("""{"op":"CONNECT", "user":"${config.master!!.user}",
"password":"${config.master!!.password}"}""")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("Master (${config.master!!.url})
and shadow (${config.shadow!!.url}) successfully connected")
            }
        }
    }
}
.join()
}
//Data structures used by REST calls in this scenario
//Structure used to read the high availability settings for a Cloud
Connector master instance
data class HAMasterConfiguration(
    var haEnabled: Boolean? = null,
    var allowedShadowHost: String? = null
)
//Structure used to read and edit the high availability settings for a Cloud
Connector shadow instance
data class HAShadowConfiguration(
    var masterPort: String,
    var masterHost: String,
    var ownHost: String? = null,
    var checkIntervalInSeconds: Int?,
    var takeoverDelayInSeconds: Int?,
    var connectTimeoutInMillis: Int?,
    var requestTimeoutInMillis: Int?
)
)

```

1.2.2.5.14.2.5 Backup And Restore Configuration

Sample Code

```

package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.BlobDataPart
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.Headers
import com.github.kittinunf.fuel.core.Method
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.result.Result
import java.io.ByteArrayInputStream
import java.io.File
/*
This example shows how to use REST APIs to perform the Backup and Restore of
the Cloud Connector configuration.
As a prerequisite you have a stable Cloud Connector configuration and you
want to save the backup for later use.

The configuration details can be found in scenario.json.
*/
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
certificate.

```

```

//So for this demonstration use case we need to deactivate all trust
checks.
disableTrustChecks()
//Use 'Connection: close' header, to make stateless communication more
efficient
FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
//Add output of cURL commands for revision
//FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
//Load configuration from property file
val config = loadScenarioConfiguration()
//Create the backup configuration of the cloud connector "master".
Fuel.post("${config.master!!.url}/api/v1/configuration/backup")
    .header(Headers.CONTENT_TYPE, "application/json")
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .body("""{"password":"my-very-secret-password"}""")
    .response { _, _, result ->
        when (result) {
            is Result.Failure -> processRequestError(result.error)
            is Result.Success ->
File("backup.zip").writeBytes(result.get())
        }
    }
    .join()
//Restore the backup configuration if some fatal accidental changes
should be reverted
val formData = listOf("password" to "my-very-secret-password")
Fuel.upload("${config.master!!.url}/api/v1/configuration/backup",
Method.PUT, formData)
    .add(BlobDataPart(ByteArrayInputStream(File("backup.zip").readBytes())
, name = "backup"))
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .response { _, _, result ->
        when (result) {
            is Result.Failure -> processRequestError(result.error)
            is Result.Success -> println("Backup configuration
successfully restored in (${config.master!!.url}) ")
        }
    }
    .join()
}

```

1.2.2.5.14.2.6 Solution Management Integration

↔ Sample Code

```

package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.jsonBody
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
import java.io.File
/*
This example shows how to use REST APIs to configure the integration with
SAP solution management.
In most cases it should be only necessary to pass a boolean flag in order to
enable or

```

disable solution management integration. It is expected that SAP host agent is already installed on the host as prerequisite for solution management integration.

This example executes requests against master and shadow instances. The shadow instance is optional. Ignore the requests to shadow instance if there is only a master instance in your environment.

The configuration details for master and shadow instances can be found in scenario.json.

```
*/
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
    certificate.
    //So for this demonstration use case we need to deactivate all trust
    checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
    efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
    //Load configuration from property file
    val config = loadScenarioConfiguration()
    //First we check the current configuration of solution management
    Fuel.get("${config.master!!.url}/api/v1/configuration/connector/
    solutionManagement")
        .authentication().basic(config.master!!.user,
    config.master!!.password)
        .responseObject<SolutionManagementConfiguration> { _, _, result ->
            when (result) {
                is Result.Success -> println("response: ${result.get()}")
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
    //Configure the hostAgent path on the shadow instance
    //Invoking this API on a shadow instance changes only the configuration.
    //Only when invoking it on a master instance it also activates solution
    management integration.
    //Note: This step is not required if host agent is installed at the
    default location
    Fuel.post("${config.shadow!!.url}/api/v1/configuration/connector/
    solutionManagement")
        .authentication().basic(config.shadow!!.user,
    config.shadow!!.password)
        .jsonBody(SolutionManagementConfiguration(hostAgentPath = "/path/to/
    hostAgent"))
        .response { _, _, result ->
            when (result) {
                is Result.Success -> println("new path to host agent was set")
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
    //Turns on the solution management integration.
    //Note: this operation is possible only on the master instance.
    //In case of an HA setup, the flag enabled is propagated to the shadow
    automatically.
    //Note: optionally you can set here the new path for host agent and
    enable DSR.
    Fuel.post("${config.master!!.url}/api/v1/configuration/connector/
    solutionManagement")
        .authentication().basic(config.master!!.user,
    config.master!!.password)
        .jsonBody(SolutionManagementConfiguration())
        .response { _, _, result ->
```

```

        when (result) {
            is Result.Success -> println("solution management is
activated")
            is Result.Failure -> processRequestError(result.error)
        }
    }.join()
    //Turns off the solution management integration.
    //Note: this operation is possible only on the master instance.
    //In case of an HA setup, the flag not enabled is propagated to the
shadow automatically
    Fuel.delete("${config.master!!.url}/api/v1/configuration/connector/
solutionManagement")
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .response { _, _, result ->
            when (result) {
                is Result.Success -> println("solution management is
deactivated")
                is Result.Failure -> processRequestError(result.error)
            }
        }
    }.join()
    //Registration file with LMDB model can be downloaded by following
request.
    //This file can be used for a manual upload to solution management.
    //If solution management integration is active,the updates are triggered
by configuration changes automatically.
    Fuel.get("${config.master!!.url}/api/v1/configuration/connector/
solutionManagement/registrationFile")
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .response { _, _, result ->
            when (result) {
                is Result.Success ->
File("lmdbModel.zip").writeBytes(result.get())
                is Result.Failure -> processRequestError(result.error)
            }
        }
    }.join()
}
//Data structures used by REST calls in this scenario
//Nullable properties are optional
data class SolutionManagementConfiguration(
    var hostAgentPath: String? = null,
    var enabled: Boolean? = null,
    var dsrEnabled: Boolean? = null
)

```

1.2.2.6 Using Service Channels

Configure Cloud Connector service channels to connect your on-premise network to specific services on SAP BTP or to an ABAP Cloud system.

Context

Cloud Connector service channels provide access from an external network to specific services on SAP BTP, or to an ABAP Cloud system. The called services are not exposed to direct access from the Internet. The Cloud Connector ensures that the connection is always available and communication is secured.

Service Channel Type	Description
RFC Connection to an ABAP Cloud system	The service channel for RFC supports calls from on-premise systems to an ABAP Cloud system using RFC.
K8s Cluster (TCP connection to a SAP BTP Kubernetes cluster)	Service channel to establish a connection to a service in a Kubernetes cluster on SAP BTP.

Next Steps

[Configure a Service Channel for RFC \[page 611\]](#)

[Configure a Service Channel for a Kubernetes Cluster \[page 613\]](#)

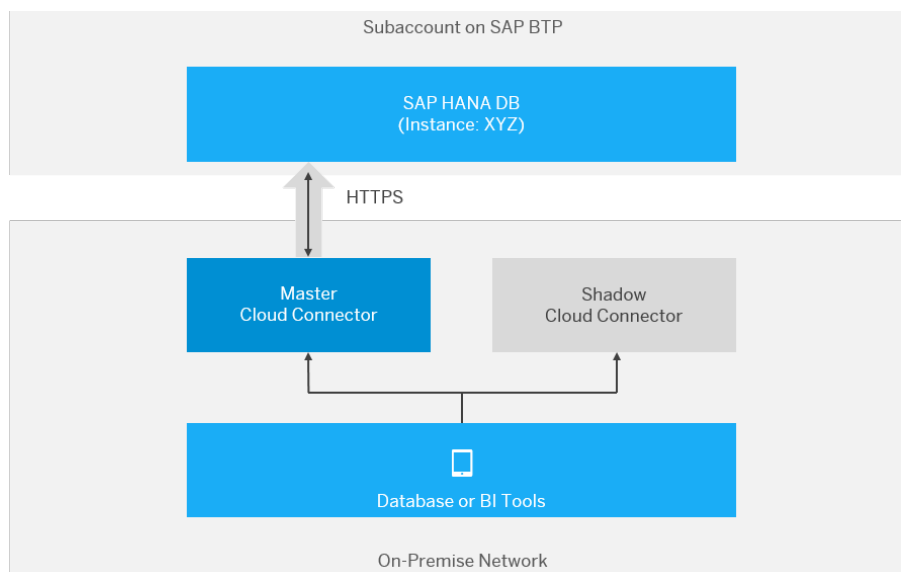
Related Information

[Service Channels: Port Overview \[page 615\]](#)

1.2.2.6.1 Connect DB Tools to SAP HANA via Service Channels

Context

You can connect database, BI, or replication tools running in on-premise network to an SAP HANA database on SAP BTP using service channels of the Cloud Connector. You can also use the high availability support of the Cloud Connector on a database connection. The picture below shows the landscape in such a scenario.



Follow the steps below to set up failover support, configure a service channel, and connect on-premise DB tools via JDBC or ODBC to the SAP HANA database.

- For more information on using SAP HANA instances, see [Using an SAP HANA XS Database System](#)
- For the connection string via ODBC you need a corresponding database user and password (see step 4 below). See also: [Creating Database Users](#).
- Find detailed information on failover support in the *SAP HANA Administration Guide: Configuring Clients for Failover*.

Note

This link points to the latest release of *SAP HANA Administration Guide*. Refer to the [SAP BTP Release Notes](#) to find out which SAP HANA SPS is supported by SAP BTP. Find the list of guides for earlier releases in the *Related Links* section below.

Procedure

1. To establish a highly available connection to one or multiple SAP HANA instances in the cloud, we recommend that you make use of the failover support of the Cloud Connector. Set up a master and a shadow instance. See [Install a Failover Instance for High Availability \[page 652\]](#).
2. In the master instance, configure a service channel to the SAP HANA database of the SAP BTP subaccount to which you want to connect. If, for example, the chosen HANA instance is 01, the port of the service channel is 30115. See also [Configure a Service Channel for an SAP HANA Database](#).
3. Connect on-premise DB tools via JDBC to the SAP HANA database by using the following connection string:

Example:

```
jdbc:sap://<cloud-connector-master-host>:30115;<cloud-connector-shadow-host>:30115[/?<options>]
```

The SAP HANA JDBC driver supports failover out of the box. All you need is to configure the shadow instance of the Cloud Connector as a failover server in the JDBC connection string. The different options supported in the JDBC connection string are described in: [Connect to SAP HANA via JDBC](#)

4. You can also connect on-premise DB tools via ODBC to the SAP HANA database. Use the following connection string:

```
"DRIVER=HDBODBC32;UID=<user>;PWD=<password>;SERVERNODE=<cloud-connector-master-host>:30115,<cloud-connector-shadow-host>:30115;"
```

Related Information

[Guides for earlier releases of SAP HANA](#)

1.2.2.6.2 Configure a Service Channel for RFC

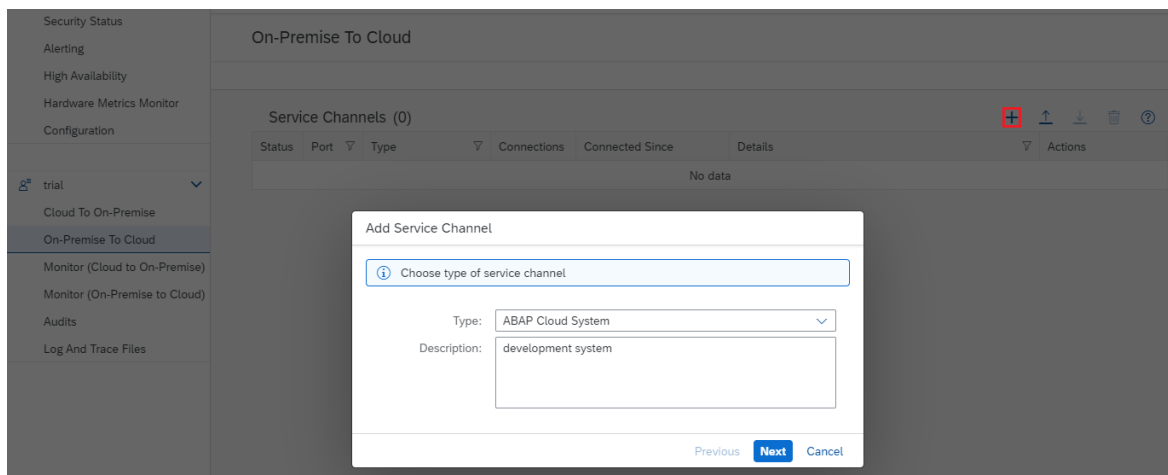
For scenarios that need to call from on-premise systems to SAP BTP ABAP environment using RFC, you can establish a connection to an ABAP Cloud tenant host. To do this, select **On-Premise to Cloud** **Service Channels** in the Cloud Connector.

Prerequisites

- When using the default connectivity setup with the **Cloud Foundry** subaccount in which the system has been provisioned, you can use a service channel without additional configuration, as long as the system is a single-tenant system.
- When using connectivity via a **Neo** subaccount, you must create a communication arrangement for the scenario SAP_COM_0200. For more information, see [Create a Communication Arrangement for Cloud Connector Integration](#) (documentation for ABAP environment on SAP BTP).

Procedure

1. From your subaccount menu, choose *On Premise To Cloud*.
2. Choose the *Add (+)* icon.



3. In the *Add Service Channel* dialog, select **ABAP Cloud System** from the drop-down list of supported channel types.
4. Optionally, provide a *Description* that explains what the ABAP Cloud service channel is used for.
5. Choose *Next*. The *ABAP Cloud System* dialog opens.
6. Enter the `<ABAP Cloud Cloud Tenant Host>` that you want to connect to.

Note

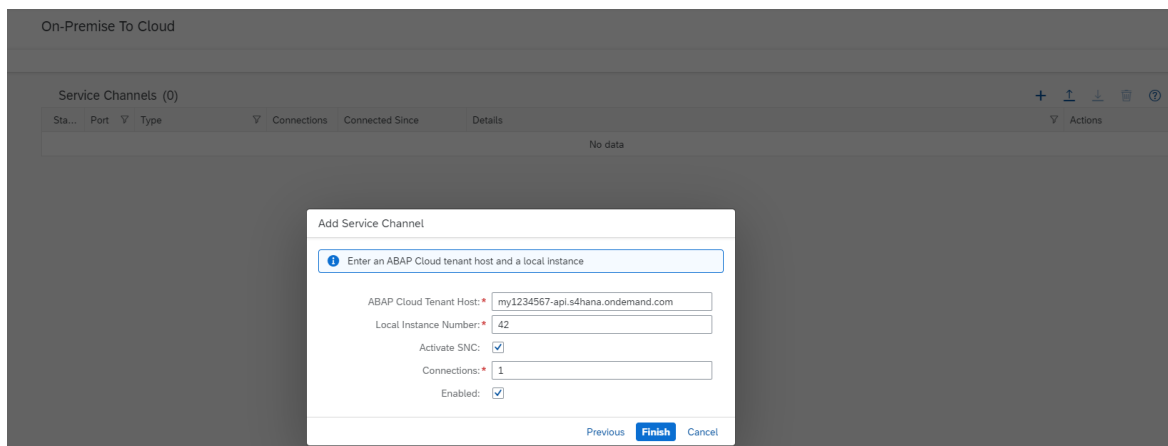
For SAP BTP ABAP environment, the tenant host is `<serviceinstanceguid>.abap.<region>.hana.ondemand.com` (note that this is not the frontend address with the `abap-web` subdomain). The region is, for example, **eu10** or **us10**.

7. In the same dialog window, define the `<Local Instance Number>` under which the ABAP Cloud system is reachable for the client systems. You can enter any instance number for which the port is not used yet on the Cloud Connector host. The port numbers result from the following pattern: **33<LocalInstanceNumber>**, for activated SNC (*Secure Network Connection*) **48<LocalInstanceNumber>**.
8. Use the checkbox *Activate SNC* to specify if the opened service channel port should listen to and terminate incoming SNC RFC connections instead of plain RFC connections. The SNC configuration used is the same as in section [Initial Configuration \(RFC\) \[page 398\]](#). In case of issues, you can use the trace *Cloud Connector loggers*, and also the *SNC payload* and *CPIC* traces. For more information on traces, see [Troubleshooting \[page 694\]](#).

Note

This SNC option is only supported for ABAP-based clients, not for any RFC connectors as JCo, NCo or NW RFC SDK. For these scenarios, please use the WebSocket RFC protocol without the Cloud Connector.

9. In the same dialog window, leave *Enabled* selected to establish the channel immediately after choosing *Finish*. Unselect it if you don't want to establish the channel immediately.



10. Choose *Finish*.

Note

When addressing an ABAP Cloud system in a **destination configuration**, you must enter the *Cloud Connector host* as application server host. As instance number, specify the `<Local Instance Number>` that you configured for the service channel. As user, you must provide the *business user* name but not the technical user name associated with the same.

Note

When using an RFC service channel in a high availability setup, you can put a TCP load balancer between the client ABAP system and the Cloud Connector master and shadow instances. In this case, you must configure the load balancer in the destination. The load balancer routes the request to the instance that is currently in the master role, either by simply trying one instance after the other or by checking regularly, which of the instances currently has the master role, and routing to this instance only.

1.2.2.6.3 Configure a Service Channel for a Kubernetes Cluster

Create a service channel to establish a connection to a service in a Kubernetes cluster in SAP BTP that is not directly exposed to external access.

Follow the steps below to establish a service channel for a Kubernetes cluster (K8s cluster).

Prerequisites

You have a service running in a Kubernetes cluster that is connected to your subaccount.

Procedure

1. Choose **► On-Premise to Cloud ► Service Channels ►** from your subaccount menu.
2. Choose the *Add* button.

Add Service Channel

i Choose type of service channel

Type:

Description:

Previous **Next** Cancel

3. In the *Add Service Channel* dialog, select **K8s Cluster** from the drop-down list of supported channel types.
4. Optionally, provide a *Description* that explains what the Kubernetes cluster service channel is used for.
5. Choose *Next*. The *K8s Cluster* dialog opens.
6. Specify the host of the Kubernetes cluster and the host providing the service inside the Kubernetes cluster.
7. Choose the *<Local Port>* and the number of *<Connections>*. You can enter any port that is not used yet.
8. Leave the *Enabled* option selected to establish the channel immediately after clicking *Save*, or deselect it if the channel should not be established yet.

Add Service Channel

i Enter a K8s Cluster and Service

K8s Cluster Host: *

K8s Service Host: *

Local Port: *

Connections: *

Enabled:

Previous
Finish
Cancel

9. When you are done, choose *Finish*.

i Note

If you enter the complete service string [`<protocol>://<host>[:<port>]/<service.host>`] into the *K8s Cluster Host* field, it is automatically split into the fields *K8s Cluster Host* and *K8s Service Host*.

Next Steps

Once you have established a service channel to the Kubernetes cluster, you can connect your client application by accessing `<Cloud_connector_host>:<local_port>`.

1.2.2.6.4 Service Channels: Port Overview

A service channel overview lets you see the details of all service channels that are used by a Cloud Connector installation.

The service channel port overview lists all service channels that are configured in the Cloud Connector. It lets you see at a glance, which server ports are used by a Cloud Connector installation.

In addition, you can find the following information about each service channel:

- Status (enabled, disabled, disconnected)
- Service channel type (ABAP Cloud System, Kubernetes Cluster)
- Assigned subaccount (display name)

- Details (for example, the assigned ABAP Cloud tenant host)

From the *Actions* column, you can switch directly to the *On-Premise To Cloud* section of the corresponding subaccount and edit the selected service channel.

To find the overview list, choose *Connector* from the navigation menu and go to section *Service Channels Overview*:

The screenshot displays the SAP Cloud Connector Administration interface. The top navigation bar includes the SAP logo, 'Cloud Connector Administration', and the user 'Administrator'. The left sidebar shows a navigation menu with options like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. The main content area is titled 'Connector' and includes a 'Select Subaccount' dropdown and 'Cross-Subaccount' options. Below this, there are buttons for '+ Add Subaccount', 'Backup', and 'Restore'. The 'Connector Overview' section shows details for a connector with ID 'ABCDEFABCDEF12345678901234567890', local name 'scc.mycompany.corp', and local IP '10.11.12.13'. It also displays security status as 'Low risk', high availability as 'High', and a 'Disconnected' status with '1' alert. The 'Subaccount Dashboard (3)' table lists three subaccounts with their status, subaccount ID, display name, location ID, region, and actions. The 'Service Channels Overview (3)' table lists three service channels with their status, port, type, subaccount, details, and actions.

Status	Subaccount	Display Name	Location ID	Region	Actions
🟢	819986f5-460d-4a36-b...	qual	EMEA	Europe (Frankfurt) - AWS	🔗 🛠️ 🗑️ ➡️
🟡	ab1220dc-5bfb-45d1-9...	trial		Trial	🔗 🛠️ 🗑️ ➡️
🟢	ztglf9vgj9	Virtual Machine Test	EMEA	Trial	🔗 🛠️ 🗑️ ➡️

Status	Port	Type	Subaccount	Details	Actions
🟢	1042	Virtual Machine	Virtual Machine Test	myVirtualMachine	🔗 ➡️
🟢	3332	ABAP Cloud System	qual	my312345-api.s4hana.ondemand.com	🔗 ➡️
🔴	31615	HANA Database	trial	abcd	🔗 ➡️

The *filter buttons* above the overview table let you filter the shown service channels based on their status. You can select all service channels, all enabled ones, all disabled ones, or all service channels for which enabling has failed.

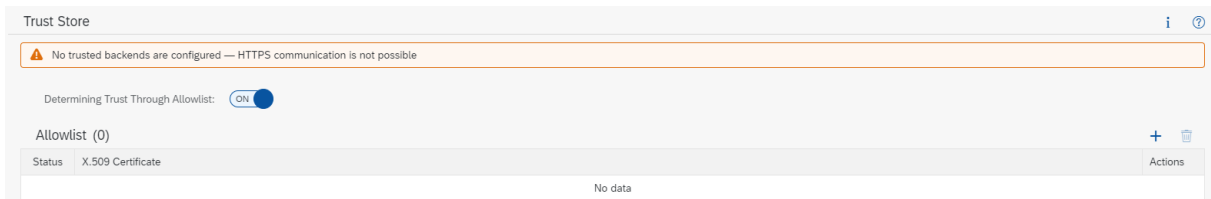
1.2.2.7 Configure Trust

Set up a trust store for on-premise systems in the Cloud Connector.

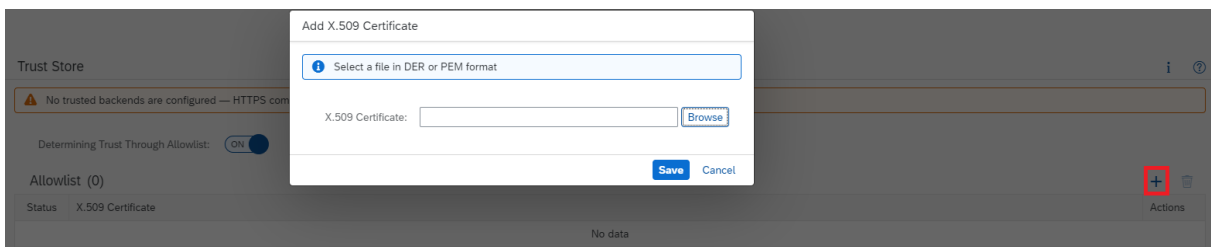
Configure the Trust Store (as of Version 2.15)

To configure the trust store, choose *Configuration* from the main menu, and go to tab *On Premise*, section *Trust Store*.

By default, the Cloud Connector **does not trust any** on-premise system when connecting to it via TLS:



To enable secured backend communication, you must add trusted certificate authorities (CAs) to the allowlist. Any TLS server certificate that has been issued by one of those CAs, will be considered trusted. If the CA that has issued a concrete server certificate is not contained in the trust store, the server will be considered untrusted and the connection will fail.



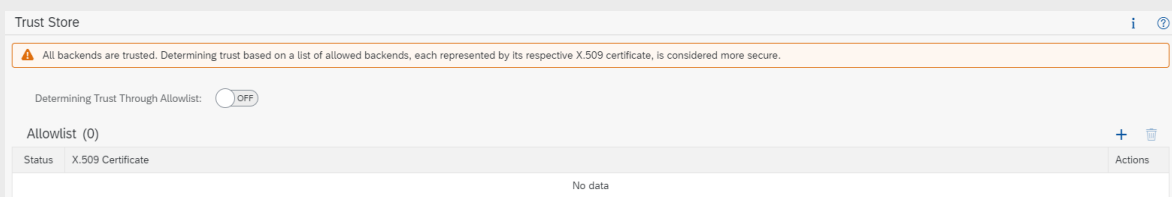
Note

You must provide the CA's X.509 certificates in DER or PEM format.

Caution

If you don't want to specify explicit CAs for trust, but rather **trust all certificates used by your backends**, you can switch off the trust store. In this case, the allowlist is ignored. However, this is considered less secure, since all server certificates are trusted and the issuing CA is not checked.

As a result, certain attacks on the hop between Cloud Connector and internal systems can be performed more easily. Therefore, **we strongly recommend that you don't do this in productive installations.**



Back to *Tasks*

1.2.2.8 Configure Domain Mappings for Cookies

Context

Some HTTP servers return cookies that contain a `domain` attribute. For subsequent requests, HTTP clients should send these cookies to machines that have host names in the specified domain.

For example, if the client receives a cookie like the following:

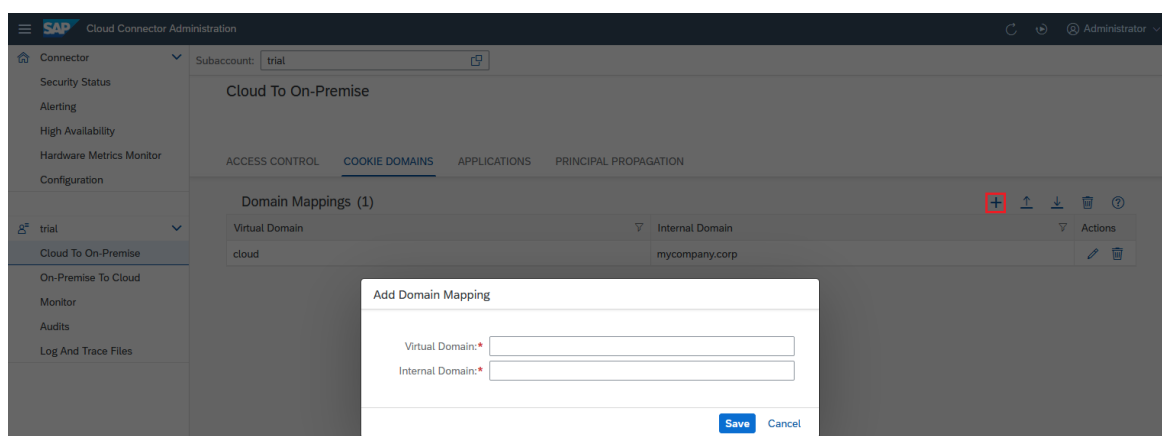
```
Set-Cookie: cookie-field=some-value; domain=mycompany.corp; path=...; ...
```

It returns the cookie in follow-up requests to all hosts like `ecc60.mycompany.corp`, `crm40.mycompany.corp`, and so on, if the other attributes like `path` and `attribute` require it.

However, in a Cloud Connector setup between a client and a Web server, this may lead to problems. For example, assume that you have defined a virtual host `sales-system.cloud` and mapped it to the internal host name `ecc60.mycompany.corp`. The client "thinks" it is sending an HTTP request to the host name `sales-system.cloud`, while the Web server, unaware of the above host name mapping, sets a cookie for the domain `mycompany.corp`. The client does not know this domain name and thus, for the next request to that Web server, doesn't attach the cookie, which it should do. The procedure below prevents this problem.

Procedure

1. From your subaccount menu, choose *Cloud To On-Premise*, and go to the *Cookie Domains* tab.
2. Choose *Add*.
3. Enter `cloud` as the virtual domain, and your company name as the internal domain.
4. Choose *Save*.



The Cloud Connector checks the Web server's response for `Set-Cookie` headers. If it finds one with an attribute `domain=intranet.corp`, it replaces it with `domain=sales.cloud` before returning the HTTP response to the client. Then, the client recognizes the domain name, and for the next request

against [www1.sales.cloud](#) it attaches the cookie, which then successfully arrives at the server on [machine1.intranet.corp](#).

Note

Some Web servers use a syntax such as `domain=.intranet.corp` (RFC 2109), even though the newer RFC 6265 recommends using the notation without a dot.

Note

The value of the domain attribute may be a simple host name, in which case no extra domain mapping is necessary on the Cloud Connector. If the server sets a cookie with `domain=machine1.intranet.corp`, the Cloud Connector automatically reverses the **mapping** [machine1.intranet.corp](#) to [www1.sales.cloud](#) and replaces the cookie domain accordingly.

Related Information

[Configure Access Control \[page 456\]](#)

1.2.2.9 Configure Solution Management Integration

Activate Solution Management reporting in the Cloud Connector.

If you want to monitor the Cloud Connector with the SAP Solution Manager, you can install a host agent on the machine of the Cloud Connector and register the Cloud Connector on your system.

Prerequisites

- You have installed the SAP Diagnostics Agent and SAP Host Agent on the Cloud Connector host and connected them to the SAP Solution Manager. The RPM on Linux ensures that the host agent configuration is adjusted and that user groups are setup correctly.
For more details about the host agent and diagnostics agent, see [SAP Host Agent](#) and the SCN Wiki [SAP Solution Manager Setup/Managed System Checklist](#).
See also SAP notes [2607632](#) (*SAP Solution Manager 7.2 - Managed System Configuration for SAP Cloud Connector*) and [1018839](#) (*Registering in the System Landscape Directory using sldreg*). For consulting, contact your local SAP partner.

Note

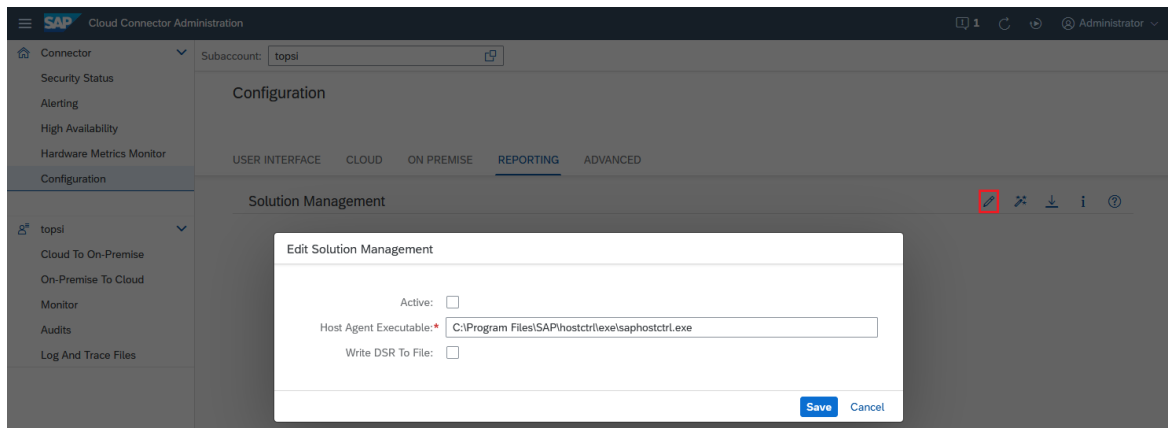
Linux OS: if you installed the host agent after installing the Cloud Connector, you can execute `enableSolMan.sh` in the installation directory to adjust the host agent configuration and user group setup. This action requires root permission.

- The SAP Solution Manager must be of release 7.2 SP06 or higher.

Procedure

To enable the integration, do the following:

1. From the Cloud Connector main menu, choose **Configuration** > **Reporting**. In section **Solution Management** of the **Reporting** tab, select **Edit**.



2. Select the **Active** checkbox.
3. In the field **<Host Agent>**, specify the location of the host agent as filepath.
4. If you want to store the reporting results (*Dynamic Statistical Records*), select **Write DSR To File**.
5. Choose **Save**.

Note

To download the registration file *lmdbModel.xml*, choose the icon **Download registration file** from the **Reporting** tab.

Related Information

[Monitoring \[page 660\]](#)

1.2.2.10 Configure Advanced Connectivity

Adapt connectivity settings that control the throughput and HTTP connectivity to on-premise systems.

Tunnel Connections

If required, you can adjust the following parameters for the communication tunnel by changing their default values:

- Application Tunnel Connections (default: 1)

Note

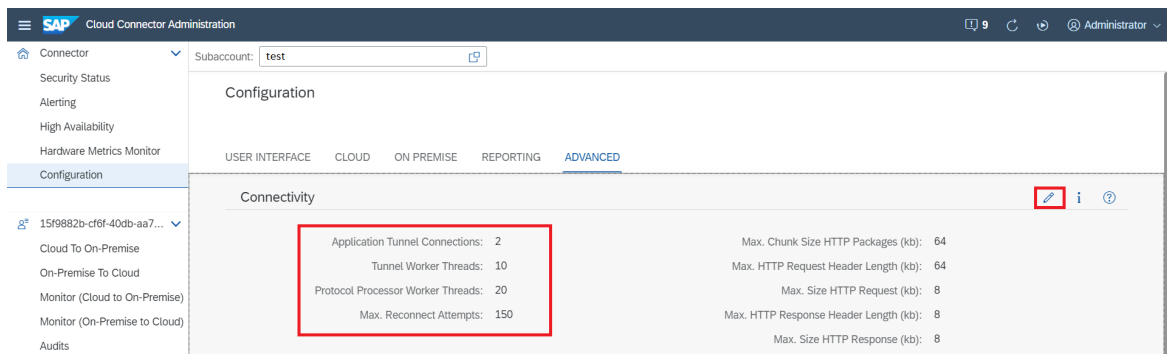
This parameter specifies the default value for the maximal number of tunnel connections *per application*. The value must be higher than 0.

- Tunnel Worker Threads (default: 10)
- Protocol Processor Worker Threads (default: 20)

For detailed information on connection configuration requirements, see [Configuration Setup \[page 373\]](#).

To change the parameter values, do the following:

1. From the Cloud Connector main menu, choose **Configuration > Advanced**. In section *Connectivity*, select *Edit*.



2. In the *Edit Connectivity Settings* dialog, change the parameter values as required.
3. Choose *Save*.

Additionally, you can specify the number of allowed tunnel connections for each application that you have specified as a [trusted application \[page 423\]](#).

Note

If you don't change the value for a trusted application, it keeps the default setting specified above. If you change the value, it may be higher or lower than the default and must be higher than 0.

HTTP Connectivity

The Cloud Connector also allows to set some sensible parameters controlling the HTTP connectivity to on-premise systems.

⚠ Caution

Do not change these parameters unless you are absolutely sure that changes are indispensable.

Configuration

USER INTERFACE CLOUD ON PREMISE REPORTING **ADVANCED**

Connectivity

Application Tunnel Connections:	1	Max. Chunk Size HTTP Packages (kb):	64
Tunnel Worker Threads:	10	Max. HTTP Request Header Length (kb):	64
Protocol Processor Worker Threads:	20	Max. Size HTTP Request (kb):	8
Max. Reconnect Attempts:	150	Max. HTTP Response Header Length (kb):	8
		Max. Size HTTP Response (kb):	8

Find a brief description of these critical configuration parameters below:

Parameter	Description	Default Value
Max. Chunk Size HTTP Packages (kb)	Max. size of chunks transmitted in HTTP streaming. The chunk size affects the throughput of HTTP communication.	64kb
Max. HTTP Request Header Length (kb)	Max. allowed size of HTTP request headers. Headers containing authentication information like SAML or JWT could require this size.	64kb
Max. Size HTTP Request (kb)	Size for the request line of HTTP request. HTTP Body is not included.	8kb
Max. HTTP Response Header Length (kb)	Max. allowed size of HTTP response headers.	8kb
Max. Size HTTP Response (kb)	Size for the response line of the HTTP response. HTTP Body is not included.	8kb

1.2.2.11 Configure the Java VM

Adapt the JVM settings that control memory management.

If required, you can adjust the following parameters for the Java VM by changing their default values:

- Initial Heap Size (default: 1024 MB)

- Maximal Heap Size (default: 1024 MB)
- Maximal Direct Memory (default: 2 GB)

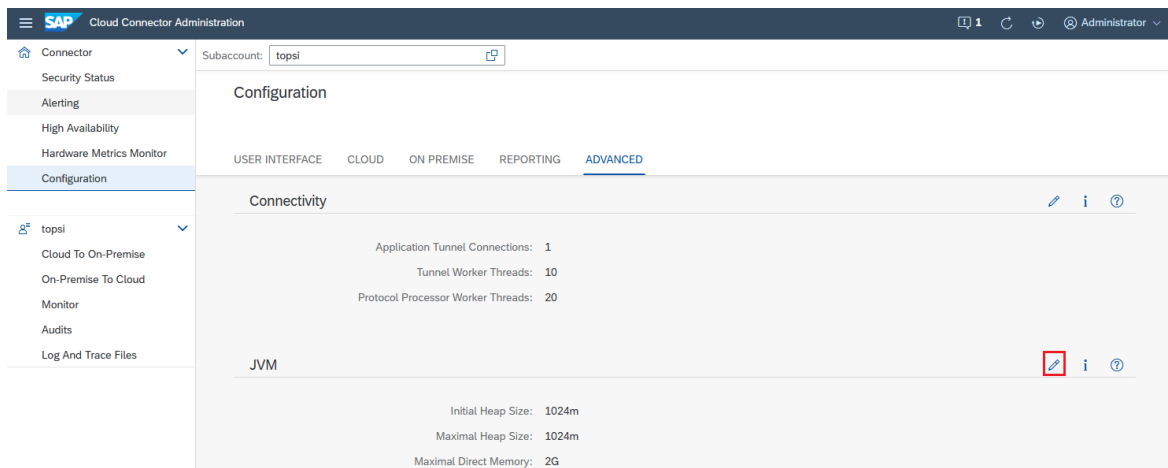
Note

A restart is required when changing JVM settings.

We recommend that you set the initial heap size equal to the maximal heap size, to avoid memory fragmentation.

To change the parameter values, do the following:

1. From the Cloud Connector main menu, choose **Configuration** > **Advanced**. In section **JVM**, select **Edit**.



2. In the **Edit JVM Settings** dialog, change the parameter values as required.
3. Choose **Save**.

Related Information

[Sizing for the Master Instance \[page 370\]](#)

1.2.2.12 Configuration Backup

You can backup and restore your Cloud Connector configuration.

To backup or restore your Cloud Connector configuration, do the following:

1. From the Cloud Connector main menu, choose **Connector**.



2. To backup or restore your configuration, choose the respective icon in the upper right corner of the screen.
 1. To backup your configuration, enter and repeat a password in the *Backup* dialog and choose *Backup*.

📄 Note

An archive containing a snapshot of the current Cloud Connector configuration is created and downloaded by your browser. For security reasons, some files are encrypted, using the password provided for the backup procedure.

→ Tip

You can use this archive to restore the current state of the same installation or to move the current configuration to a new Cloud Connector installation, if the original instance can no longer be used.

You can restore the archive on a Cloud Connector instance

- With the same or higher version than the original one.
- Running on a different operating system.

However, you cannot restore it on an *older* Cloud Connector version.

2. To restore your configuration, enter the required *Archive Password* and the *Login Password* of the currently logged-in administrator user in the *Restore from Archive* dialog and choose *Restore*.

⚠ Caution

The restore action overwrites the current configuration of the Cloud Connector. Its current state and settings will be lost permanently unless you have created another backup before doing the restore operation. Upon successfully restoring the configuration backup, the Cloud Connector restarts automatically. All active sessions are then terminated.

📄 Note

- The `props.ini` file is treated in a special way. If the file from the backup archive differs from the one that is used in the current installation, it will be placed next to the original one as `props.ini.restored`. If you want to use the `props.ini.restored` file, replace the existing `props.ini` file by the restored one on OS level and restart the Cloud Connector afterwards.
- All custom path settings from the configuration backup archive, which are not valid or operable anymore, will be automatically reset to their respective default values upon restore operation. For example, this will be the case if a configured directory or file path from the backup archive does not exist or is not accessible anymore.
- Restored certificates from older backup archives might have expired and must be replaced with valid ones to get all configured scenarios to work again.
- If restoring a configuration backup archive on a host that is different from the original one, the restored UI certificate might not be issued for the new hostname, domain or IP address, and therefore would require a replacement with a matching UI server certificate as well.
- If you are using an external SNC library, neither the library itself is part of the configuration backup archive, nor any configuration that this library would load from its own external storage. As a result, the SNC configuration must be set up again after restoring a configuration backup archive on a different host.
- The same applies to used own CA certificates which are stored in the JVM's managed trust store, for example, if Email alerting or LDAP has been set up this way, with own certificates for

establishing a secure communication. These externally stored CA certificates would also not be part of the configuration backup archive and must be imported into the local JVM's trust store manually.

⚠ Caution

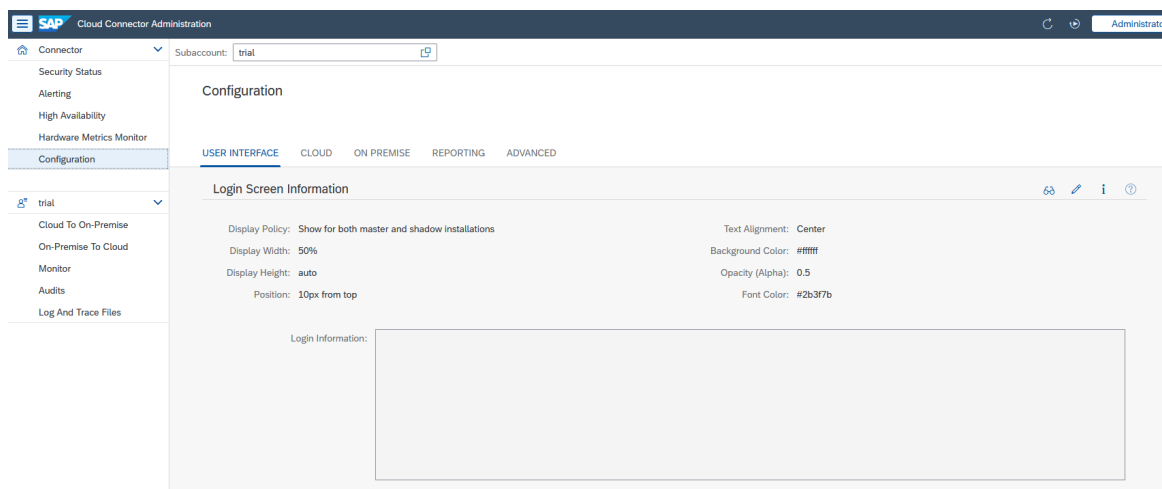
Do not run multiple Cloud Connector instances with the same configuration simultaneously. The feature to restore a configuration backup archive on another Cloud Connector installation is not supposed to be used for cloning purposes, but only for supporting the move of a Cloud Connector instance from one host to another. After restoring a configuration backup archive on a different Cloud Connector installation, you must stop the original Cloud Connector instance *immediately*. Otherwise, running two or more instances with the same configuration will cause issues. Such a cloned setup is not supported.

1.2.2.13 Configure Login Screen Information

Add additional information to the login screen and configure its appearance.

To configure the login screen information, proceed as follows:

1. Go to ► **Configuration** ► **User Interface** ► (Shadow instance: ► **Shadow Configuration** ► **User Interface** ►) and press the **Edit** button in section **Login Screen Information** (at the bottom of the screen).



As a result, the following dialog is opened:

Edit Login Information

Display Policy

- Show for both master and shadow installations Show only if this is a master installation
 Show only if this is a shadow installation Do not show at all

Login Display Properties



Display Width:	<input type="text" value="50%"/>	Background Color:	<input type="text" value="#ffffff"/>
Display Height:	<input type="text" value="auto"/>	Opacity (Alpha):	<input type="text" value="0.5"/>
Text Alignment:	<input type="text" value="Center"/>	Font Color:	<input type="text" value="#2b3f7b"/>
Position:	<input type="text" value="10px"/>	<input checked="" type="radio"/> from top	<input type="radio"/> from bottom

Login Information



Enter an HTML fragment to be displayed as login information. Restrictions apply. Compliance with these restrictions will be checked when saving. Consult the documentation for details.

Save Cancel

- In section *Display Policy*, select a display policy for the login screen information. The display policy decides whether the information is shown if the instance is assuming the role mentioned in the policy.
- In section *Display Properties*, specify your preferred display properties (appearance and position):
 - The login information is displayed in a box with rounded corners. You can specify its width and height in pixels (unit px) or as a percentage (unit %).

Note

Omitting any value triggers the default or auto behavior.

- For the width, the default behavior is equivalent to 100%.
- For the height, the default behavior sets the height to a value that accommodates the login information (that is, the given HTML fragment). For extensive information we therefore

recommend that you limit the height to a suitable pixel or percentage value to induce scrolling, and to prevent the box from growing beyond a reasonable height.

- When customizing the **background color** of the box, the **opacity** (of the background color), **font color**, or **text alignment**, then the section *Login Information* automatically switches to preview mode. That way you can follow live the changes made to the appearance of the box and of the information displayed inside it.

Note

You can hide the box and to show only the text of the login information by choosing an opacity value of **0** (opacity is the opposite of transparency. *No* opacity means *complete* transparency).

- You can **position the box** containing the login information at the top or bottom of the login page. To do this, set the field `<Position>` to the corresponding pixel or percentage value.
4. Enter the information to be displayed in section *Login Information*. The information must be supplied as an HTML fragment. There is a limited number of tags that can be used. Attributes available for these tags are subject to restrictions.

Available Tag	Attribute Restriction
p	Only attribute style is permitted, with property <code>text-align</code> set to a valid value (left , right , center , or justified)
ul	No attributes allowed
ol	No attributes allowed
li	No attributes allowed
br	No attributes allowed
h1	No attributes allowed
h2	No attributes allowed
h3	No attributes allowed
i	No attributes allowed
b	No attributes allowed
a	Must have exactly two attributes: <ul style="list-style-type: none"> <code>href</code> (its value must be a <URL> with protocol http or https) <code>target</code> (its value must be "_blank")

HTML syntax checking is strict. Attribute values must be enclosed by double quotes. Missing or unmatched opening or closing tags are not permitted.

Note

Tag `br` does not require a closing tag as there cannot be any inner HTML.

1.2.3 Administration

Learn more about operating the Cloud Connector, using its administration tools and optimizing its functions.

Topic	Description
Exchange UI Certificates in the Administration UI [page 629]	By default, the Cloud Connector includes a self-signed UI certificate. It is used to encrypt the communication between the browser-based user interface and the Cloud Connector itself. For security reasons, however, you should replace this certificate with your own one to let the browser accept the certificate without security warnings.
Logon to the Cloud Connector via Client Certificate [page 631]	Switch from default logon to client certificate logon to access the Cloud Connector.
Configure Named Cloud Connector Users [page 633]	If you operate an LDAP server in your system landscape, you can configure the Cloud Connector to use the named users who are available on the LDAP server instead of the default Cloud Connector users.
High Availability Setup [page 651]	The Cloud Connector lets you install a redundant (shadow) instance, which monitors the main (master) instance.
Change the UI Port [page 657]	Use the changeport tool (Cloud Connector version 2.6.0+) to change the port for the Cloud Connector administration UI. .
Connect and Disconnect a Cloud Subaccount [page 658]	As a Cloud Connector administrator, you can connect the Cloud Connector to (and disconnect it from) the configured cloud subaccount.
Secure the Activation of Traffic Traces [page 659]	Tracing of network traffic data may contain business critical information or security sensitive data. You can implement a "four-eyes" (double check) principle to protect your traces (Cloud Connector version 1.3.2+).
Monitoring [page 660]	Use various views to monitor the activities and state of the Cloud Connector.
Alerting [page 687]	Configure the Cloud Connector to send email alerts whenever critical situations occur that may prevent it from operating.
Audit Logging [page 690]	Use the auditor tool to view and manage audit log information (Cloud Connector version 2.2+).
Troubleshooting [page 694]	Information about monitoring the state of open tunnel connections in the Cloud Connector. Display different types of logs and traces that can help you troubleshoot connection problems.

Topic	Description
Process Guidelines for Hybrid Scenarios [page 699]	How to manage a hybrid scenario, in which applications running on SAP BTP require access to on-premise systems using the Cloud Connector.
Configuring Backup [page 702]	Find an overview of backup procedures for the Cloud Connector.
Secure Store [page 703]	Reduce the size of the Cloud Connector Secure Store.

1.2.3.1 Exchange UI Certificates in the Administration UI

By default, the Cloud Connector includes a self-signed UI certificate. It is used to encrypt the communication between the browser-based user interface and the Cloud Connector itself. For security reasons, however, you should replace this certificate with your own one to let the browser accept the certificate without security warnings.

Procedure

Master Instance

1. From the main menu, choose *Configuration* and go to the *User Interface* tab.
2. In the *UI Certificate* section, start a certificate signing request procedure by choosing the icon *Generate a Certificate Signing Request*.
3. In the pop-up *Generate CSR*, specify a key size and a subject fitting to your host name. For host matching, you should use the available names within the `subjectAlternativeName` (SAN) extension, see [RFC 2818](#) and <https://www.chromestatus.com/feature/4981025180483584>. A check verifies whether the host matches one of the entries in the SAN extension. In section **Subject Alternative Names**, you can add additional values by pressing the *Add* button. Choose one or more of the following SAN types and provide the matching values:
 - DNS: a specific host name (for example, `www.sap.com`) or a wildcard hostname (for example, `*.sap.com`).
 - IP: an IPv4 or IPv6 address.
 - [RFC822](#): an example for this type of value is a simple email address: for example, `donotreply@sap.com`.
 - URI: a URI for which the certificate should be valid.

Generate CSR

Key Size

2048 Bits
 4096 Bits

Subject DN

Common Name (CN): *

E-Mail Address (EMAIL):

Locality (L):

Organizational Unit (OU):

Organization (O):

State or Province (ST):

Country (C):

Subject Alternative Names

+

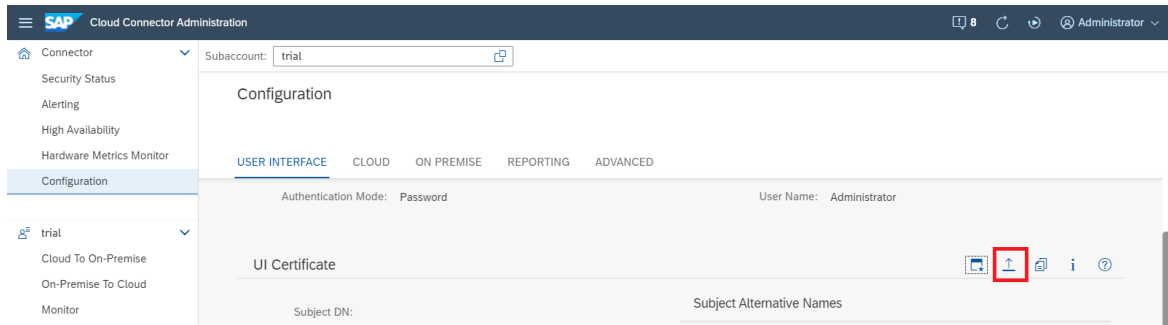
Type	Value	Actions
<input type="text" value="RFC822"/>	<input type="text" value="ownerscc@company.com"/>	
<input type="text" value="DNS"/>	<input type="text" value="*.internal.corp"/>	

- Press *Generate*.
- You are prompted to save the signing request in a file. The content of the file is the signing request in PEM format.
The signing request must be provided to a Certificate Authority (CA) - either one within your company or another one you trust. The CA signs the request and the returned response should be stored in a file.

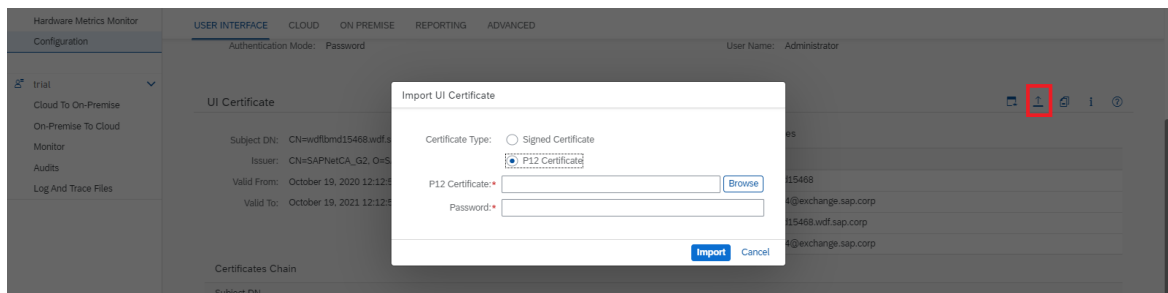
Note

The response should be either an X.509 certificate or a PKCS#7 in PEM format.

- To import the signing response, choose the *Upload* icon.



You can also upload an existing PKCS#12 certificate directly (instead of generating a CSR).



7. Select *Browse* to locate the file and then choose the *Import* button.
8. Review the certificate details that are displayed.
9. Restart the Cloud Connector to activate the new certificate.

Shadow Instance

In a High Availability setup, perform the same operation on the shadow instance.

⚠ Caution

UI certificates are used for the secure communication between master and shadow instances. Replacing the UI certificate breaks the trust relationship and communication between master and shadow is not possible anymore.

Please disconnect the shadow instance when you are going to replace UI certificate(s). Once the certificate update is done, connect the shadow instance again. You will be forced to enter user and password again to establish the trust relationship between master and shadow instances.

1.2.3.2 Logon to the Cloud Connector via Client Certificate

Switch from default logon to client certificate logon to access the Cloud Connector.

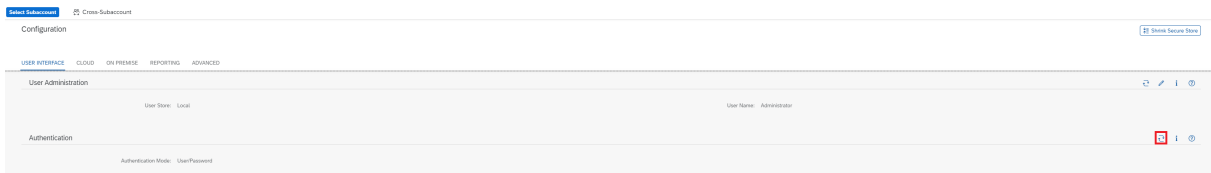
⚠ Restriction

Limitation for Cloud Connector version 2.16.0: This feature does not work if *high availability* setup is active. A Cloud Connector shadow instance cannot connect due to missing trust with error: *There is no trust with Cloud Connector on https://<host>:<port>. Reset shadow configuration and try to connect shadow again.*

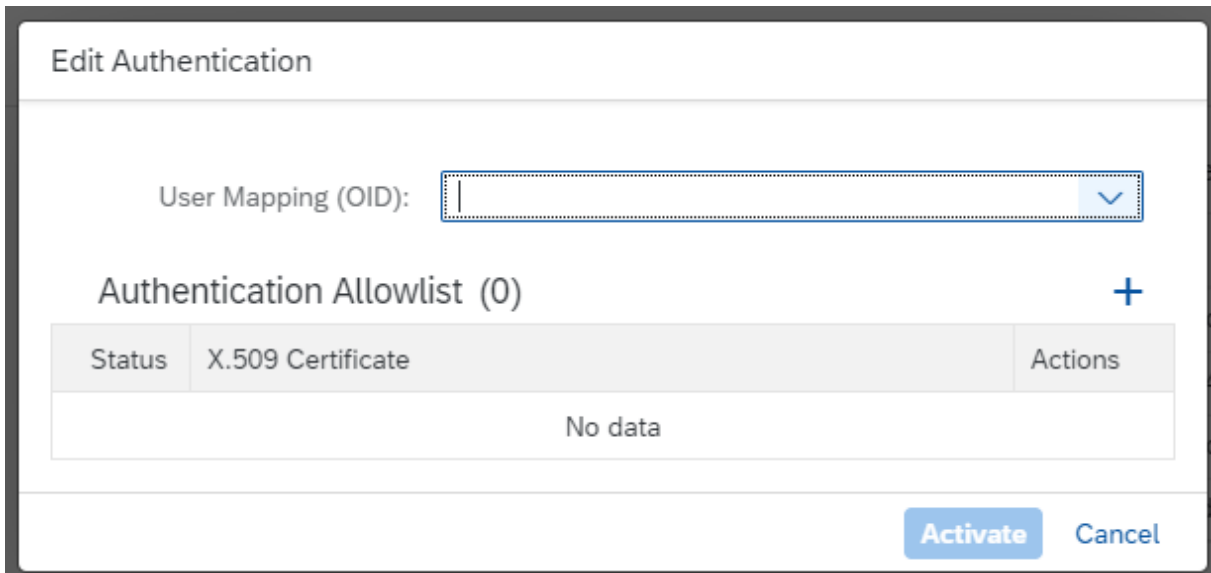
⚠ Restriction

Configuring the logon with a certificate is not allowed for SAP-operated Cloud Connectors, for example in the context of *Enterprise Cloud Services* or *S/4HANA Private Cloud Edition*.

Instead of authenticating with user and password as configured by default (see [Initial Configuration \[page 388\]](#)), you can switch to client certificate authentication and logon. To do so, choose *Configuration* (or *Shadow Configuration*) from the main menu, and tab *User Interface*, section *Authentication*.



In the next prompt, specify an object identifier (OID) to extract the user name from subject of any given client certificate.



You must add at least one CA certificate to the *Authentication Allowlist*. These certificates are used by Cloud Connector to validate an incoming client certificate. Typically, these are the root or issuer certificates of eligible client certificates, or the client certificates themselves if they are self-signed.

After activation, a restart is required.

The login works as follows:

1. (Prerequisite): Your browser (or your REST client) must provide a suitable client certificate. In particular, the user extracted from such a client certificate as per the selected *User Mapping (OID)* must be a member of the current user store (local or LDAP).
2. When accessing the Cloud Connector, an mTLS handshake is performed, that is, the Cloud Connector checks your selected client certificate against the CAs from the authentication allowlist.

Caution

If your client certificate is not self-signed, you must provide (one of) the issuer certificates in the *authentication allowlist*, not the client certificate from your browser.

3. If trust can be established, the Cloud Connector extracts the user from the subject as per the specified user mapping OID. This user will be logged on if it can be found in the current user store. If LDAP is used, the respective roles are checked and assigned.

If you select, for example, *CN* as OID, and have a client certificate with a subject containing *CN=Administrator*, the user *Administrator* is used after mutual trust was established.

1.2.3.3 Configure Named Cloud Connector Users

Set up LDAP-based user management for the Cloud Connector.

LDAP-Based User Management

We recommend that you configure LDAP-based user management for the Cloud Connector to allow only named administrator users to log on to the administration UI.

This guarantees traceability of the Cloud Connector configuration changes via the Cloud Connector audit log. If you use the default and built-in `Administrator` user, you cannot identify the actual person or persons who perform configuration changes. Also, you will not be able to use different types of user groups.

Configuration

If you have an LDAP server in your landscape, you can configure the Cloud Connector to authenticate Cloud Connector users against the LDAP server.

Valid users or user groups must be assigned to one of the following roles:

- Administrator users: `admin` or `sccadmin`
- Display users: `sccdisplay` or `sccmonitoring`

Note

The role `sccmonitoring` provides access to the monitoring APIs, and is particularly used by the SAP Solution Manager infrastructure, see [Monitoring APIs \[page 670\]](#).

- Support users: `sccsupport`

Alternatively, you can define custom role names for each of these user groups, see: [Use LDAP for User Administration \[page 634\]](#).

Once configured, the default Cloud Connector `Administrator` user becomes inactive and can no longer be used to log on to the Cloud Connector.

Related Information

[Use LDAP for User Administration \[page 634\]](#)

[Logon to the Cloud Connector via Client Certificate \[page 631\]](#)

1.2.3.3.1 Use LDAP for User Administration

You can use LDAP (Lightweight Directory Access Protocol) to manage Cloud Connector users and authentication.

After installation, the Cloud Connector uses file-based user management by default. Alternatively, the Cloud Connector also supports LDAP-based user management. If you operate an LDAP server in your landscape, you can configure the Cloud Connector to use the LDAP user base.

If LDAP authentication is active, you can assign users or user groups to the following default roles:

User Role	Authorization
<code>sccadmin</code> or <code>admin</code>	Administrate the Cloud Connector (all CRUD operations).
<code>sccsubadmin</code>	<ul style="list-style-type: none">• Manage all subaccount-related settings.• Perform support-related tasks like setting trace levels or creating a thread dump. Access to common settings for all subaccounts, like system certificate settings, is permitted.
<code>sccdisplay</code>	Access the Cloud Connector administration UI in read-only mode.
<code>sccsupport</code>	<ul style="list-style-type: none">• Access the Cloud Connector administration UI in read-only mode.• Perform support-related tasks like setting trace levels or creating a thread dump.

User Role

sccmonitoring

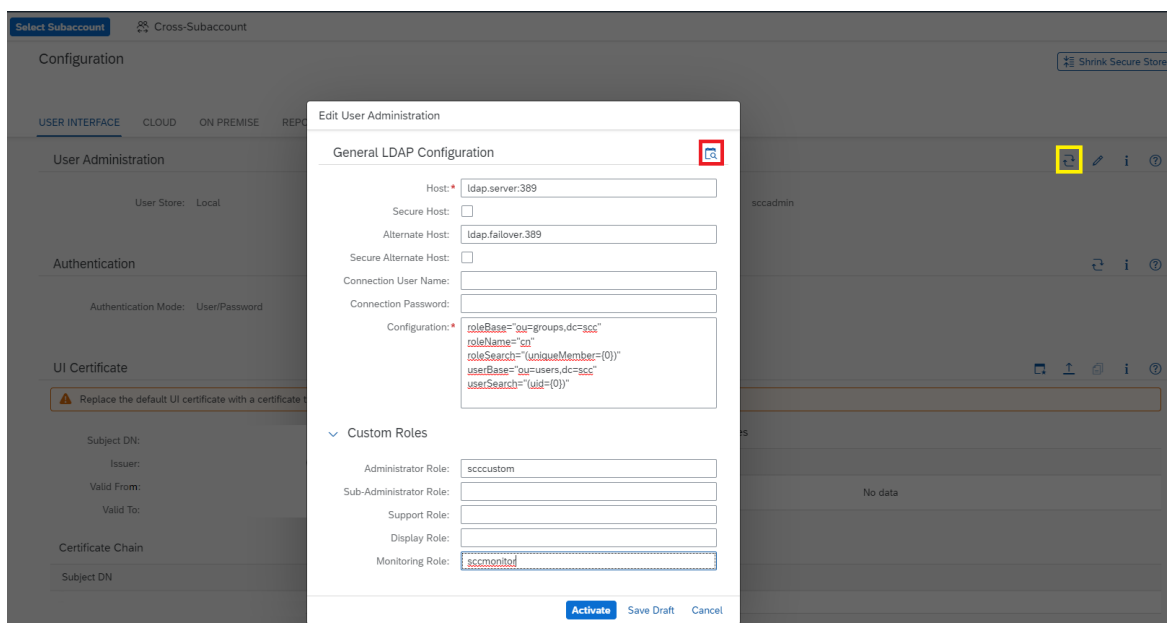
Authorization

Provides access to the monitoring APIs, and is particularly used by the SAP Solution Manager infrastructure, see [Monitoring APIs \[page 670\]](#).

Group membership is checked by the Cloud Connector.

Setting LDAP Authentication

1. From the main menu, choose *Configuration* and go to the *User Interface* tab.
2. From the *User Administration* section, choose *Switch to LDAP*.



3. (Optional) To save intermediate adoptions of the LDAP configuration, choose *Save Draft*. This lets you store the changes in the Cloud Connector without activation.
4. Usually, the LDAP server lists users in an LDAP node and user groups in another node. In this case, you can use the following template for LDAP configuration. Copy the template into the configuration text area:

```
roleBase="ou=groups,dc=scc"  
roleName="cn"  
roleSearch="(uniqueMember={0})"  
userBase="ou=users,dc=scc"  
userSearch="(uid={0})"
```

Change the `<ou>` and `<dc>` fields in `userBase` and `roleBase`, according to the configuration on your LDAP server, or use some other LDAP query.

Note

The configuration depends on your specific LDAP server. For details, contact your LDAP administrator.

5. Provide the LDAP server's host and port (port **389** is used by default) in the `<Host>` field. To use the secure protocol variant LDAPS based on TLS, select *Secure*.
6. Provide a failover LDAP server's host and port (port **389** is used by default) in the `<Alternate Host>` field. To use the secure protocol variant LDAPS based on TLS, select `<Secure Alternate Host>`.
7. (Optional) Depending on your LDAP server configuration you may need to specify the `<Connection User Name>` and its `<Connection Password>`. LDAP Servers supporting anonymous binding ignore these parameters.
8. (Optional) To use your own role names, you can customize the default role names in the *Custom Roles* section. If no custom role is provided, the Cloud Connector checks permissions for the corresponding default role name:
 - `<Administrator Role>` (default: `sccadmin`)
 - `<Support Role>` (default: `sccsupport`)
 - `<Display Role>` (default: `sccdisplay`)
 - `<Monitoring Role>` (default: `sccmonitoring`)
9. (Optional) Before activating the LDAP authentication, you can execute an authentication test by choosing the *Test LDAP Configuration* button. In the pop-up dialog, you must specify user name and password of a user who is allowed to logon after activating the configuration. The check verifies if authentication would be successful or not.

Note

We strongly recommend that you perform an authentication test. If authentication should fail, login is not possible anymore. The test dialog also provides a test protocol, which could be helpful for troubleshooting.

For more information about how to set up LDAP authentication, see tomcat.apache.org/tomcat-8.5-doc/realm-howto.html .

Note

To find a list of all supported attributes, see https://tomcat.apache.org/tomcat-8.5-doc/config/realm.html#JNDI_Directory_Realm_-_org.apache.catalina.realm.JNDIRealm .

You can also configure LDAP authentication on the shadow instance in a high availability setup (master and shadow). From the main menu of the shadow instance, select *Shadow Configuration*, go to tab *User Interface*, and check the *User Administration* section.

Note

If you are using LDAP together with a high availability setup, you cannot use the configuration option `userPattern`. Instead, use a combination of `userSearch`, `userSubtree` and `userBase`.

Caution

An LDAP connection over TLS can cause TLS errors if the LDAP server uses a certificate that is not signed by a trusted CA. If you cannot use a certificate signed by a trusted CA, you must set up the trust relationship manually, that is, import the public part of the issuer certificate to the JDK's trust storage.

Usually, the `cacerts` file inside the `java` directory (`jre/lib/security/cacerts`) is used for trust storage. To import the certificate, you can use `keytool`:

```
keytool -import -storepass changeit -file <certificate used by LDAP server> -keystore cacerts -alias <e.g. LDAP_xyz>
```

For more information, see also <https://docs.oracle.com/cd/E19830-01/819-4712/ablqw/index.html>.

10. After finishing the configuration, choose *Activate*. Immediately after activating the LDAP configuration you must restart the Cloud Connector server, which invalidates the current browser session. Refresh the browser and logon to the Cloud Connector again, using the credentials configured at the LDAP server.
11. To switch back to file-based user management, choose the *Switch* icon in section *User Administration* again.

Note

If you have set up an LDAP configuration incorrectly, you may not be able to logon to the Cloud Connector again. In this case, adjust the Cloud Connector configuration to use the file-based user store again *without the administration UI*. For more information, see the next section.

Switching Back to File-Based User Store without the Administration UI

If your LDAP settings do not work as expected, you can use the `useFileUserStore` tool, provided with Cloud Connector version 2.8.0 and higher, to revert back to the file-based user store:

1. Change to the installation directory of the Cloud Connector and enter the following command:
 - **Microsoft Windows:** `useFileUserStore`
 - **Linux, Mac OS:** `./useFileUserStore.sh`
2. Restart the Cloud Connector to activate the file-based user store.

For versions older than 2.8.0, you must manually edit the configuration files.

Depending on your operating system, the configuration file is located at:

- **Microsoft Windows OS:**
`<install_dir>\config_master\org.eclipse.gemini.web.tomcat\default-server.xml`
- **Linux OS:** `/opt/sap/scc/config_master/org.eclipse.gemini.web.tomcat/default-server.xml`
- **Mac OS X:** `/opt/sap/scc/config_master/org.eclipse.gemini.web.tomcat/default-server.xml`

1. Replace the `Realm` section with the following:

```
<Realm className="org.apache.catalina.realm.LockOutRealm">
  <Realm className="org.apache.catalina.realm.CombinedRealm">
    <Realm
      X509UsernameRetrieverClassName="com.sap.scc.tomcat.utils.SccX509SubjectDnRetriever"
      className="org.apache.catalina.realm.UserDatabaseRealm"
      digest="SHA-256" resourceName="UserDatabase"/>
    <Realm
      X509UsernameRetrieverClassName="com.sap.scc.tomcat.utils.SccX509SubjectDnRetri
```

```
ever" className="org.apache.catalina.realm.UserDatabaseRealm" digest="SHA-1"
resourceName="UserDatabase" />
</Realm>
</Realm>
```

2. Restart the Cloud Connector service:

- **Microsoft Windows OS:** Open the Windows [Services](#) console and restart the cloud connector service.
- **Linux OS:** Execute
 - System V init distributions: `service scc_daemon restart`
 - Systemd distributions: `systemctl restart scc_daemon`
- **Mac OS X:** Not applicable because no daemon exists (for Mac OS X, only a portable variant is available).

Related Information

[LDAP Configuration: Best Practices \[page 638\]](#)

1.2.3.3.1.1 LDAP Configuration: Best Practices

Get background information on LDAP configuration for the Cloud Connector.

[Introduction \[page 638\]](#)

[Connect to the LDAP Server \[page 639\]](#)

[TLS Issues \[page 640\]](#)

[Authentication \[page 641\]](#)

[User Selection \[page 643\]](#)

[User Roles \[page 647\]](#)

[Relationship between User and Group \[page 648\]](#)

[Custom User Roles \[page 650\]](#)

[Additional Notes \[page 651\]](#)

Introduction

Using an LDAP server for user management allows seamless integration of the Cloud Connector into the on-premise environment. It requires some configuration that must match the setup on your LDAP server, and therefore can't be generated automatically.

The configuration parameters are common for various products and mostly well known.

The apache tomcat project, which is used as underlying technology by the Cloud Connector, provides an excellent tutorial: tomcat.apache.org/tomcat-8.5-doc/realms-howto.html . It explains the LDAP configuration parameters and considers various LDAP directory setups, including their specific configuration.

However, some aspects may raise questions. For this reason, we show you how to configure LDAP and verify LDAP configuration, providing useful background information in this topic.

A basic understanding of LDAP and tomcat's how-to guide is a prerequisite. As help tool, we are using the *ldapsearch* utility. You can use any LDAP client for this procedure.

Back to [Top \[page 638\]](#)

Connect to the LDAP Server

In a first step, you must establish a connection to the LDAP server. Like an HTTP connection, the connection to LDAP can be secure (via TLS) or plain. It points to a host and port. The address looks like this:

- TLS connection: `ldaps://<ldap.server.in.your.company>:<numeric port>`
- Plain connection: `ldap://<ldap.server.in.your.company>:<numeric port>`

Sample Code

```
ldapsearch -H ldap://<ldaphost>:<port>
```

The return value is `-1` if the address is not reachable. Before you go ahead, you need to know the address of your LDAP server. As soon as the *ldapsearch* utility returns a value other than `-1`, the address of the LDAP server is correct. More precisely, it indicates only that there is a server listening on this port, which is supposed to be the LDAP server.

Once the address is known, you can test the connection in the Cloud Connector. Enter the address and add a dummy configuration, for example, `x="x"`, to outwit the check. Then choose the test icon in the upper right corner. For a valid address, the LDAP configuration test in the Cloud Connector reports the following:

The screenshot shows the 'Edit Authentication' configuration page for an LDAP server. A red error message at the top states: 'LDAP server authentication failed →Details'. Below this is the 'General LDAP Configuration' section with fields for Host, Secure Host, Alternate Host, Secure Alternate Host, Connection User Name, Connection Password, and Configuration (set to 'x="x"'). At the bottom are 'Activate', 'Save Draft', and 'Cancel' buttons. On the right, the 'Test Results' panel shows 'Host: Reachable' and 'Authentication: Failed'. The details of the failure are listed as follows:

- Connecting to URL ldap://wdfbmd15468.wdf.sap.corp:10389
- Exception performing authentication. Retrying... throwing exception javax.naming.AuthenticationException: [LDAP: error code 49 - INVALID_CREDENTIALS: Bind failed: ERR_229 Cannot authenticate user]
- Connecting to URL ldap://wdfbmd15468.wdf.sap.corp:10389
- Exception performing authentication throwing exception javax.naming.AuthenticationException: [LDAP: error code 49 - INVALID_CREDENTIALS: Bind failed: ERR_229 Cannot authenticate user]
- Returning null principal.
- Authentication for user c failed

Below the details, it says 'What You Can Do: Verify connection user and password' and provides a link for support: 'Download test results including LDAP configuration!'.

The message: *Connecting to URL ldap://<ldaphost>:<port>; Exception performing authentication* indicates that your LDAP server requires user and password for queries.

Back to [Top \[page 638\]](#)

TLS Issues

LDAP connection over TLS will run into TLS errors if the LDAP server uses an "untrusted" certificate. This could be a self-signed certificate or a certificate signed by a generally untrusted authority.

If you cannot use a trusted certificate on your LDAP server, you must import the public part of the issuer certificate to the JDK's trust storage. See the JDK documentation how to do that.

Usually, the trust storage location is `cacerts` inside the `java` directory (`jre/lib/security/cacerts`). You can use the `keytool` utility for import.

Sample Code

```
keytool -import -storepass changeit -file <certificate used by LDAP server>
-keystore cacerts -alias <e.g. LDAP_xyz>
```

See also: [Working with Certificates and SSL](#) (Java documentation).

Back to [Top \[page 638\]](#)

Authentication

If the address of the LDAP server is correct and the Cloud Connector can establish a connection, you can proceed with the next step: the authentication by the LDAP directory.

The LDAP server may require authentication or not (anonymous connection), before a query can be executed. Authentication is done by user and password, specified by the `connectionName` and `connectionPassword` properties.

Anonymous access is sufficient in most cases and provides the same level of security. However, you have to deal with the existing setup on the LDAP server.

Note

The LDAP user is not the same user that is later used to logon on to the Cloud Connector. It is a specific user, which has permissions to query the LDAP directory. It can be stored in an LDAP directory separated from other users. We recommend that you specify the fully-qualified user name like `"uid=admin,ou=system"`.

Additionally, *MS Active Directory* allows authentication for `user@domain`, for example, `ldapUser@company.local`.

To verify the values, let's first check the authentication with `ldapsearch`:

Sample Code

```
ldapsearch -D "uid=admin,ou=system" -w <password> -H ldap://<ldaphost>:<port>
-b "x=x"
```

Note

We added a non-existing user base `-b "x=x"` to prevent long output.

The only thing to check here is, if authentication is ok. If it is not, LDAP returns `INVALID_CREDENTIALS: Bind failed`:

Sample Code

```
ldapsearch -D "uid=admin,ou=system" -w <wrong password> -H ldap://
<ldaphost>:<port> -b "x=x"
ldap_bind: Invalid credentials (49)
additional info: INVALID_CREDENTIALS: Bind failed: ERR_229 Cannot
authenticate user uid=admin,ou=system
```


You can perform the same test in the Cloud Connector UI:

Edit Authentication

General LDAP Configuration 🔍

*Host:

Secure Host:

Alternate Host:

Secure Alternate Host:

Connection User Name:

Connection Password:

*Configuration:

▼ **Custom Roles**

Administrator Role:

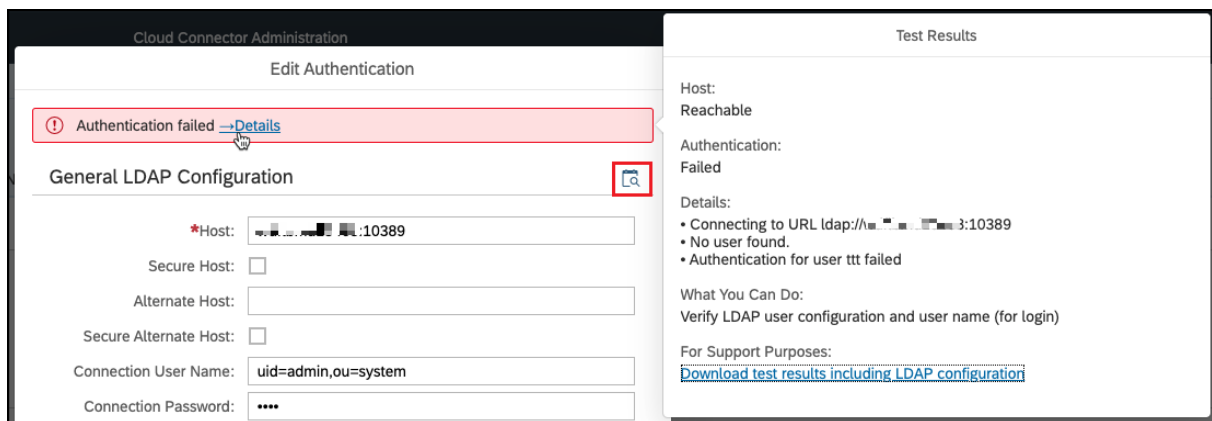
Support Role:

Display Role:

Monitoring Role:

Save Draft Activate Cancel

Choose the test icon in the upper right corner, and enter something as name and password:



Authentication failed in this check, but the connection to the LDAP server was successful.

If you get an *Exception performing authentication* message here, check the reply from the LDAP server and align your parameters until you can connect.

Back to [Top \[page 638\]](#)

User Selection

Once authentication is checked, set the root node for users. User nodes are nodes containing user details. They are located somewhere in the LDAP tree. Sometimes they are all listed under one branch (parent node), but they may also be distributed across several branches. In any case, start with one branch that contains at least one user node.

List the user nodes with *ldapsearch*:

Sample Code

```
ldapsearch -D "uid=admin,ou=system" -w <password> -H ldap://<ldaphost>:<port>
-b "ou=users,dc=scc"
```

The output contains the nodes located under the specified base. Each node looks like this:

Sample Code

```
# Thor, users, scc
dn: uid=Thor,ou=users,dc=scc
sn: SCC Administrator
cn: sccadmin
objectClass: top
objectClass: inetOrgPerson
objectClass: person
objectClass: organizationalPerson
userPassword:: ...
uid: Thor
```

Basically, every unique parameter can be used as user ID. The user *Thor* in this example could also enter its DN (distinguished name) as user name. However, this is not very user-friendly. To not upset Thor, you can define the attribute containing the user ID that is used for logon.

`userSearch` selects the attribute containing the user ID. Together with the `userBase`, the configuration looks like this now:

↔ Sample Code

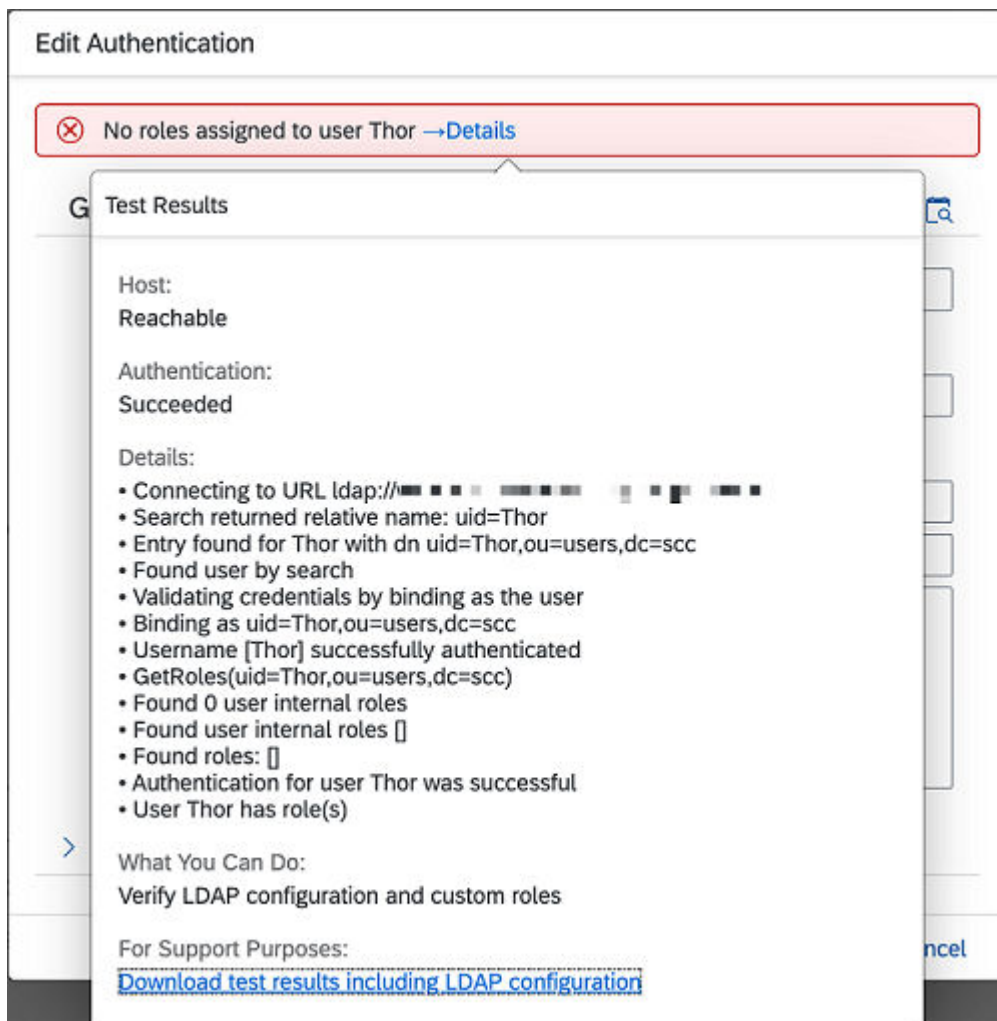
```
userBase="ou=users,dc=scc"  
userSearch="(uid={0})"
```

The corresponding LDAP selection is:

↔ Sample Code

```
ldapsearch -D "uid=admin,ou=system" -w <password> -H ldap://<ldaphost>:<port>  
-b "ou=users,dc=scc" "(&(uid=Thor))"
```

Now, we take a closer look now at the response. The user *Thor* was found, its password was successfully validated by LDAP server, but there are no specific roles selected:



Let's assume that the users are not located under the same branch in the LDAP tree. For this case, you cannot define more than one *userBase*. Instead, you can set *userBase* for the corresponding parent node, which then includes all the user branches. To achieve this, add `userSubtree="true"` to the configuration.

Taking a look at the relative name returned by search, it is `uid=Thor,ou=users` now.

Besides `uid`, you are free to use every other attribute of the user node. For example, for *Active Directory*, the preferred attribute is often `sAMAccountName`, the corresponding configuration is `userSearch="(sAMAccountName={0})"`.

Note

For *Active Directory*, it might be necessary to add `adCompat="true"` to the configuration.

As mentioned above, you can use every attribute as user ID as long as it is *unique*. To verify this, check the query result for the respective attribute with `ldapsearch`. If it contains more than one node, the test in the Cloud Connector would report *User name [<userid>] has multiple entries, No user found, Authentication for user <userid> failed*. In our test, the CN (common name) attribute is not unique and the following search returns more than one entry.

Sample Code

```
ldapsearch -D "uid=admin,ou=system" -w <password> -H ldap://<ldaphost>:<port>
-b "ou=users,dc=scc" "(&(cn=<userid>))"
```

Note

If `connectionUser` is located under `userBase` and its ID can be selected by the same `userSearch`, you can use just the user ID in the `connectionUser` field instead of a fully-qualified DN.

Back to [Top \[page 638\]](#)

User Roles

At this point, the configuration lets you establish a connection to the LDAP server and authenticate a user.

In the next step, we configure authorization, that is, the roles assigned to a user.

A role is a *group* in LDAP terms. The Cloud Connector provides the following roles: `sccadmin`, `sccsubadmin`, `sccsupport`, `sccmonitoring` and `sccdisplay`.

Most likely, your LDAP server does not define such groups. Here, the best practice is to create new groups for role assignment. Using these special groups for managing Cloud Connector users lets administrators easily grant permissions to the relevant users. For example, only users with administrator permissions would be added to the `sccadmin` group. Like this, you avoid side effects, and you can increase the security and stability levels of the Cloud Connector.

Reuse of already existing groups is also possible. Set these group names as custom roles in the Cloud Connector's LDAP configuration. However, keep in mind that every user in the existing group will automatically get permissions for the Cloud Connector. Even if at present all users in the available group should have permissions for Cloud Connector, this could cause issues at some point in the future. To avoid this, custom roles should not be used for reuse of existing groups on your LDAP server. The main purpose of reused groups is to create group names that match your company's naming conventions.

Like users, also groups are represented as nodes in an LDAP tree branch. The branch where the groups are located must be configured as `roleBase`.

The `roleName` defines, which of its attributes is taken as role name. Usually, you wouldn't use the fully-qualified distinguished name as role name.

Check if `ldapsearch` can find entries for the role:

Sample Code

```
ldapsearch -D "uid=admin,ou=system" -w <password> -H ldap://<ldaphost>:<port>
-b "<roleBase>" "(&(cn=<roleName>))"
```

The screenshot displays the 'Edit Authentication' configuration window for an LDAP connection. A red error message at the top states: 'No roles assigned to user Thor ->Details'. The 'General LDAP Configuration' section includes fields for Host, Secure Host, Alternate Host, Secure Alternate Host, Connection User Name (uid=admin,ou=system), Connection Password, and Configuration (userBase="ou=users,dc=scc", userSearch="(uid={0})", roleBase="ou=groups,dc=scc", roleName="cn"). A 'Custom Roles' link is visible at the bottom left. The 'Test Results' panel on the right shows the following details:

- Host: Reachable
- Authentication: Succeeded
- Details:
 - Connecting to URL ldap://wdfbmd15468.wdf.sap.corp:10389
 - Search returned relative name: uid=Thor
 - Entry found for Thor with dn uid=Thor,ou=users,dc=scc
 - Found user by search
 - Validating credentials by binding as the user
 - Binding as uid=Thor,ou=users,dc=scc
 - Username [Thor] successfully authenticated
 - GetRoles(uid=Thor,ou=users,dc=scc)
 - Found 0 user internal roles
 - Found user internal roles []
 - Found roles: []
 - Authentication for user Thor was successful
 - User Thor has role(s)
- What You Can Do: Verify LDAP configuration and custom roles
- For Support Purposes: [Download test results including LDAP configuration](#)

When trying to define only `roleBase` and `roleName`, the test reports no roles for the specified user: *Found roles: []*. The empty brackets indicate an 'empty collection'. The reason is that the configuration does not define how users are related to the found groups. Find some background information on this in the next section.

Back to [Top \[page 638\]](#)

Relationship between User and Group

LDAP provides two ways to define the relationship between a user and its groups:

1. The group node contains one or more attributes `uniqueMember`, OR
 2. Uses one or more attributes `memberOf` in a user node.
- **Case 1:** Extend the current configuration by `roleSearch="(uniqueMember={0})"`. The Cloud Connector's test reports direct roles in this case. Roles selected by `roleSearch` are LDAP entries. Using configuration parameter `roleName`, you specify the LDAP entry attribute which will be the role name. In our example, we are using `roleName="cn"` with the following result in the test: *Found direct role cn=sccadmin,ou=groups,dc=scc -> sccadmin*. The selected value `sccadmin` is the value of the attribute `cn` located in the original LDAP entry.

- **Case 2:** Extend the current configuration by `userRoleName="memberOf"`. This is reported by the Cloud Connector's test as internal role. If no groups are defined by `memberOf`, the internal group list is empty.

→ Remember

The roles specified by an attribute of a user entry are used as is, that is, the configuration parameter `roleName` is irrelevant in this case, and the role name is just set to the attribute value.

Below, the test report selected neither internal nor direct roles:

The screenshot displays the 'Edit Authentication' configuration window and the 'Test Results' panel. In the configuration window, the 'General LDAP Configuration' section shows the following settings:

- Host: [Redacted]
- Secure Host:
- Alternate Host: [Empty]
- Secure Alternate Host:
- Connection User Name: `uid=admin,ou=system`
- Connection Password: [Redacted]
- Configuration:


```

userBase="ou=users,dc=scc"
userSearch="(uid={0})"
roleBase="ou=groups,dc=scc"
roleName="cn"
userRoleName="memberOf"
roleSearch="(uniqueMemberX={0})"

```

The 'Test Results' panel shows the following details:

- Host: Reachable
- Authentication: Succeeded
- Details:
 - Connecting to URL `ldap://wdfibmd15468.wdf.sap.corp:10389`
 - Search returned relative name: `uid=Thor`
 - Entry found for Thor with dn `uid=Thor,ou=users,dc=scc`
 - Retrieving values for attribute `memberOf`
 - Found user by search
 - Validating credentials by binding as the user
 - Binding as `uid=Thor,ou=users,dc=scc`
 - Username [Thor] successfully authenticated
 - GetRoles(`uid=Thor,ou=users,dc=scc`)
 - Found 0 user internal roles
 - Found user internal roles []
 - Found 0 direct roles
 - Found roles: []
 - Authentication for user Thor was successful
 - User Thor has role(s)

To demonstrate an empty result, the parameter `roleSearch` was set to a non-existing attribute here.

Below, the test reflecting LDAP configuration eventually reports a non-empty list of found roles, containing the role `sccadmin`:

Edit Authentication

✓ User Thor authenticated →Details

General LDAP Configuration

Host: * [redacted]

Secure Host:

Alternate Host: []

Secure Alternate Host:

Connection User Name: uid=admin,ou=system

Connection Password: ****

Configuration: *
 userBase="ou=users,dc=scc"
 userSearch="(uid={0})"
 roleBase="ou=groups,dc=scc"
 roleName="cn"
 roleSearch="(uniqueMember={0})"

> Custom Roles

Activate Save Draft Cancel

Test Results

Active Roles:
sccadmin

Details:

- Connecting to URL ldap://wdfbmd15468.wdf.sap.corp:10389
- Search returned relative name: uid=Thor
- Entry found for Thor with dn uid=Thor,ou=users,dc=scc
- Found user by search
- Validating credentials by binding as the user
- Binding as uid=Thor,ou=users,dc=scc
- Username [Thor] successfully authenticated
- GetRoles(uid=Thor,ou=users,dc=scc)
- Found 0 user internal roles
- Found user internal roles []
- Search returned relative name: cn=sccadmin
- Retrieving attribute cn
- Found 1 direct roles
- Found direct role cn=sccadmin,ou=groups,dc=scc -> sccadmin
- Found roles: [sccadmin]
- Authentication for user Thor was successful
- User Thor has role(s) sccadmin

For Support Purposes:
[Download test results including LDAP configuration!](#)

Note

Like user nodes, also group nodes on the LDAP server may be located under several branches inside the "base" branch. In this case, add the boolean attribute `roleSubtree="true"`.

Back to [Top \[page 638\]](#)

Custom User Roles

If you want to use non-default groups, you can set a group of your choice in the **Custom Roles** section. You can replace one or more roles.

Note

Keep in mind that the custom role definition replaces the standard role. So, once a custom role for *Administrator* is set, the standard one (`sccadmin`) is not effective anymore.

Back to [Top \[page 638\]](#)

Additional Notes

There are some more configuration parameters available that are out of scope here. Most LDAP configurations are covered by the parameters discussed above.

For special purposes, you may have to add to your configuration:

- `adCompat="true"`, if your LDAP server uses *MS Active Directory* and you encounter strange errors in the test report.
- `forceDnHexEscape="true"`, if your LDAP server uses *MS Active Directory* and there are non-standard characters in the DN.
- `connectionTimeout="x"`, if you want to change the default of 5s.

General Recommendations

- Don't use the `userPattern` parameter. It invalidates SSL/TLS-based authentication and high availability setup would fail.
- If the user ID has **non-standard characters**, escape them with `\nn`.
- For **back-slash**, always use `\\`.

Back to [Top \[page 638\]](#)

1.2.3.4 High Availability Setup

You can operate the Cloud Connector in a high availability mode, in which a master and a shadow instance are installed.

Task	Description
Install a Failover Instance for High Availability [page 652]	Install a redundant Cloud Connector instance (shadow) that monitors the main instance (master).
Master and Shadow Administration [page 655]	Learn how to operate master and shadow instances.

Related Information

[Connect DB Tools to SAP HANA via Service Channels \[page 609\]](#)

1.2.3.4.1 Install a Failover Instance for High Availability

The Cloud Connector lets you install a redundant instance that monitors the main instance.

Context

In a failover setup, when the main instance should go down for some reason, a redundant one can take over its role. The main instance of the Cloud Connector is called **master** and the redundant instance is called the **shadow**. The shadow has to be installed and connected to its master. During the setup of high availability, the master pushes the entire configuration to the shadow. Later on, during normal operation, the master also pushes configuration updates to the shadow. Thus, the shadow instance is kept synchronized with the master instance. The shadow pings the master regularly. If the master is not reachable for a while, the shadow tries to take over the master role and to establish the tunnel to SAP BTP.

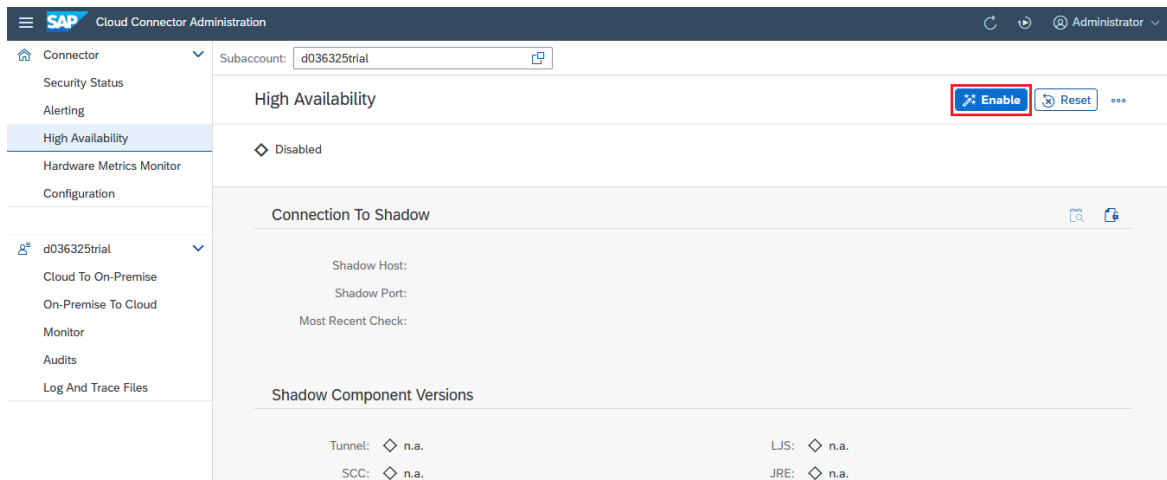
Note

For detailed information about sizing of the master and the shadow instance, see also [Sizing Recommendations \[page 369\]](#).

Procedure

Preparing the Master Instance for High Availability

1. Open the Cloud Connector UI and go to the master instance.
2. From the main menu, choose *High Availability*.
3. Choose *Enable*.



If this flag is not activated, no shadow instance can connect to this Cloud Connector. Additionally, when providing a concrete *Shadow Host*, you can ensure that only from this host a shadow instance can be connected.

⚠ Caution

Pressing the [Reset](#) button resets all high availability settings to their initial state. As a result, high availability is disabled and the shadow host is cleared. Reset only works if no shadow is connected.


Installing and Setting Up a Shadow Instance

Install the shadow instance in the same network segment as the master instance. Communication between master and shadow via proxy is not supported. The same distribution package is used for master and shadow instance.

📘 Note

If you plan to use LDAP for the user authentication on both master and shadow, make sure you configure it **before** you establish the connection from shadow to master.

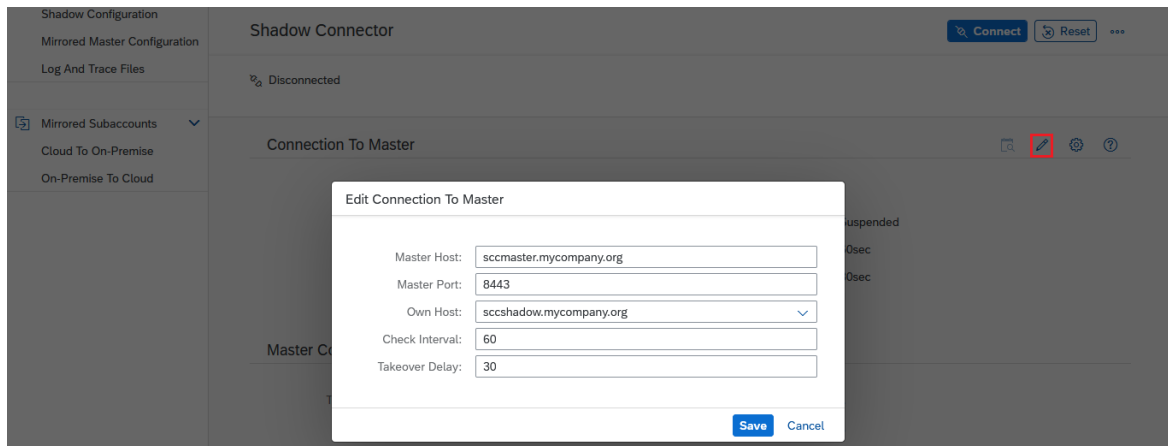
1. On first start-up of a Cloud Connector instance, a UI wizard asks you whether the current instance should be master or shadow. Choose [Shadow](#) and [Save](#):

 Save

Choose Installation Type

- Master (Primary Installation)
- Shadow (Backup Installation)

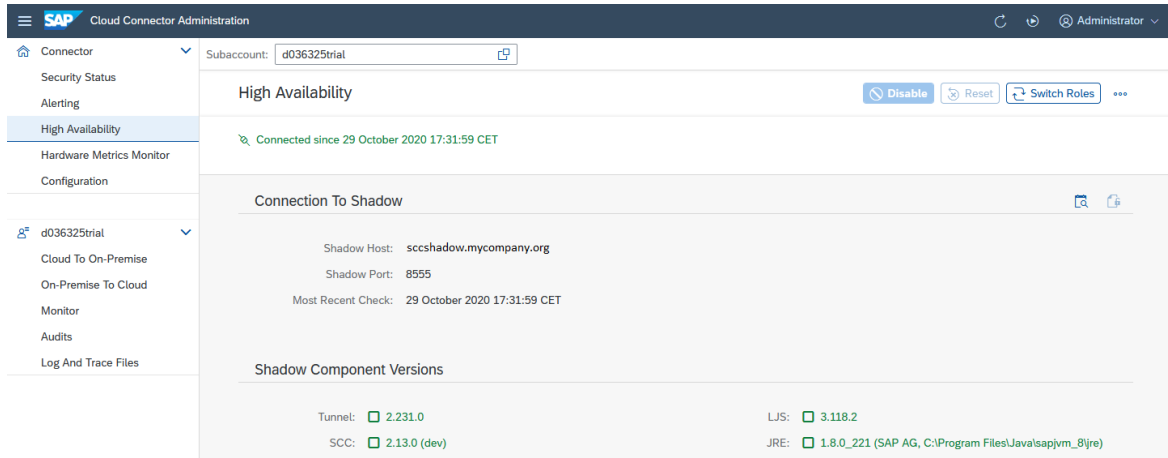
2. From the main menu, choose [Shadow Connector](#) and provide connection data for the master instance, that is, the master host and port. You can choose from the list of known host names, to use the host name under which the shadow host is visible to the master. You can specify a host name manually, if the one you want is not on the list. For the first connection, you must log on to the master instance, using the user name and password for the master instance. The master and shadow instances exchange X.509 certificates, which will be used for mutual authentication.



Note

If you want to attach the shadow instance to a different master, press the [Reset](#) button. All your high availability settings will be removed, that is, reset to their initial state. This works only if the shadow is not connected.

- Upon a successful connection, the master instance pushes the entire configuration plus some information about itself to the shadow instance. You can see this information in the UI of the shadow instance, but you can't modify it.
- The UI on the master instance shows information about the connected shadow instance. From the main menu, choose [High Availability](#):



- [High Availability](#) alert messages inform you if configuration changes have not been pushed successfully. This might happen, for example, if a temporary network failure occurs at the same time a configuration change is made. As an administrator, you can check if there is an inconsistency in the configuration data between master and shadow that could cause trouble if the shadow needs to take over. Typically, the master recognizes this situation and tries to push the configuration change at a later time automatically. If this is successful, all failure alerts are removed and replaced by a warning alert showing that there had been trouble before. These alerts are displayed in the general [Alerting](#) section. If the master doesn't recover automatically, disconnect, then reconnect the shadow, which triggers a complete configuration transfer.

Related Information

[Initial Configuration \[page 388\]](#)

[Master and Shadow Administration \[page 655\]](#)

1.2.3.4.2 Master and Shadow Administration

Administration of Shadow Instances

There are several administration activities you can perform on the shadow instance. All configuration of tunnel connections, host mappings, access rules, and so on, must be maintained on the master instance; however, you can replicate them to the shadow instance for display purposes. You may want to modify the **check interval** (time between checks of whether the master is still alive) and the **takeover delay** (time the shadow waits to see whether the master would come back online, before taking over the master role itself).

As of Cloud Connector version 2.11.2, you can configure the timeout for the connection check, by pressing the gear icon in the section *Connection To Master* of the shadow connector main page.

Edit Timeout Settings

 Validate and/or change current timeout settings

Timeout Settings

Connection Timeout (in ms):

Request Timeout (in ms):

Timeout Validation

Number of requests:

Requests:

Connection Timeouts:

Errors:

Request Timeouts:

Save

Cancel

The `<Connection Timeout>` field specifies the maximum time allowed for establishing the technical connection to the master, the `<Request Timeout>` defines the maximum time allowed for executing the check over this connection. In the *Timeout Validation* section, you can check the current settings by pressing the *Execute* button. If the timeouts are hit, you might want to increase the settings accordingly.

Keep in mind the following points:

- The log level on master and shadow instances can be different.
- Configuration for check interval and takeover delay is maintained only on the shadow instance, and is transferred to the master for display purposes.
- Audit logs are only written on the master instance and are not transferred to the shadow. However, if the shadow becomes the master for some time, the audit log is potentially distributed over both master and shadow instances.

You can use the *Reset* button to drop all the configuration information on the shadow that is related to the master, but only if the shadow is not connected to the master.

Required Configuration on the Shadow Instance

Once connected to the master, the shadow instance receives the configuration from the master instance. Yet, there are some aspects you must configure on the shadow instance separately:

- **User administration** is configured separately on master and shadow instances. Generally, it is not required to have the same configuration on both instances. In most cases, however, it is suitable to configure master and shadow in the same way.
- The **UI certificate** is not shared. Each host can have its own certificate, so you must maintain the UI certificates on master and shadow. You can use the same certificate though.
- **SNC configuration**: If secure RFC communication or principal propagation for RFC calls is used, you must configure SNC on each instance separately.

Failover Process

The shadow instance regularly checks whether the master instance is still alive. If a check fails, the shadow instance first attempts to reestablish the connection to the master instance for the time period specified by the takeover delay parameter.

- If no connection becomes possible during the takeover delay time period, the shadow tries to take over the master role. At this point, it is still possible for the master to be alive and the trouble to be caused by a network issue between the shadow and master.
The shadow instance next attempts to establish a tunnel to the given SAP BTP subaccount. If the connection attempt fails to all configured subaccounts (for whatever reason), the shadow instance remains in "shadow status", periodically pinging the master and trying to connect to the cloud, while the master is not yet reachable.
- Otherwise, if the tunnel to the cloud side can be opened, the shadow instance will take over the master role. From this moment, the shadow instance displays the UI of a master instance and allows the usual

operations of a master instance, for example, starting/stopping tunnels, modifying the configuration, and so on.

This is, in particular, also the case if the master is still alive, but the network connection between shadow and master is corrupted. This leads to a *master-master* setup, which is automatically detected on the former master once the network between the two instances recovers. The former master will then automatically relinquish its master role and assume the shadow role to get back to the desired setup.

When the original master instance restarts, it first checks whether the registered shadow instance has taken over the master role. If it has, the master registers itself as a shadow instance on the former shadow (now master) instance. Thus, the two Cloud Connector installations, in fact, have switched their roles.

Note

Only one shadow instance is supported. Any further shadow instances that attempt to connect are declined by the master instance.

The master considers a shadow as lost, if no check/ping is received from that shadow instance during a time interval that is equal to three times the check period. Only after this much time has elapsed can another shadow system register itself.

Note

On the master, you can manually trigger failover by selecting the [Switch Roles](#) button. If the shadow is available, the switch is made as expected. Even if the shadow instance cannot be reached, the role switch of the master may still be enforced. Select [Switch Roles](#) only if you are absolutely certain it is the correct action to take for your current circumstances.

Even with the active role switch, zero downtime is not guaranteed. Depending on various aspects and timings, there may be short time slots in which establishing new connections fails. When switching the role, all active requests on the master will be broken as the sockets will be closed.

1.2.3.5 Change the UI Port

Context

By default, the Cloud Connector uses port **8443** for its administration UI. If this port is blocked by another process, or if you want to change it after the installation, you can use the `changeport` tool, provided with Cloud Connector version **2.6.0** and higher.

Note

On Windows, you can also choose a different port during installation.

Procedure

1. Change to the installation directory of the Cloud Connector. To adjust the port and execute one of the following commands:

- Microsoft Windows OS:

```
changeport <desired_port>
```

- Linux OS, Mac OS X:

```
./changeport.sh <desired_port>
```

2. When you see a message stating that the port has been successfully modified, restart the Cloud Connector to activate the new port.

1.2.3.6 Connect and Disconnect a Cloud Subaccount

The major principle for the connectivity established by the Cloud Connector is that the Cloud Connector administrator should have full control over the connection to the cloud, that is, deciding if and when the Cloud Connector should be connected to the cloud, the accounts to which it should be connected, and which on-premise systems and resources should be accessible to applications of the connected subaccount.

Using the administration UI, the Cloud Connector administrator can connect and disconnect the Cloud Connector to and from the configured cloud subaccount. Once disconnected, no communication is possible, either between the cloud subaccount and the Cloud Connector, or to the internal systems. The connection state can be verified and changed by the Cloud Connector administrator on the Subaccount Dashboard tab of the UI.

The screenshot shows the SAP Cloud Connector Administration interface. The main content area is titled 'Connector' and includes a 'Subaccount: trial' dropdown. Below this is the 'Connector Overview' section, which displays the Connector ID (F98F5E60FE3211E4AD44EC790A124F3F), Local Name, Local IP, Security Status (Low risk), High Availability (Disabled), and Alerts (0). The 'Subaccount Dashboard (2)' section contains a table with the following data:

Status	Subaccount	Display Name	Location ID	Region	Actions
	trial	trial		Europe (Rot)	
	maasdemo	maasdemo		Europe (Rot)	

Note

Once the Cloud Connector is freshly installed and connected to a cloud subaccount, none of the systems in the customer network are yet accessible to the applications of the related cloud subaccount. Accessible systems and resources must be configured explicitly in the Cloud Connector one by one, see [Configure Access Control \[page 456\]](#).

A Cloud Connector instance can be connected to multiple subaccounts in the cloud. This is useful especially if you need multiple subaccounts to structure your development or to stage your cloud landscape into

development, test, and production. In this case, you can use a single Cloud Connector instance for multiple subaccounts. However, we recommend that you do not use subaccounts running in productive scenarios and subaccounts used for development or test purposes within the same Cloud Connector. You can add or delete a cloud account to or from a Cloud Connector using the [Add](#) and [Delete](#) buttons on the [Subaccount Dashboard](#) (see screenshot above).

Related Information

[Managing Subaccounts \[page 401\]](#)

1.2.3.7 Secure the Activation of Traffic Traces

For support purposes, you can trace HTTP and RFC network traffic that passes through the Cloud Connector.

Context

Traffic data may include business-critical information or security-sensitive data, such as user names, passwords, address data, credit card numbers, and so on. Thus, by activating the corresponding trace level, a Cloud Connector administrator might see data that he or she is not meant to. To prevent this behavior, implement the four-eyes principle for your operating system as described below.

Once the four-eyes principle is applied, activating a trace level that dumps traffic data will require two separate users:

- An operating system user on the machine where the Cloud Connector is installed;
- An `Administrator` user of the Cloud Connector user interface.

By assigning these roles to two different people, you can ensure that both persons are needed to activate a traffic dump.

Four-Eyes Principle for Microsoft Windows OS

1. Create a file named `writeHexDump` in `<scc_install_dir>\scc_config`. The owner of this file must be a user other than the operating system user who runs the `cloud connector` process.

Note

Usually, this file owner is the user which is specified in the [Log On](#) tab in the properties of the `cloud connector` service (in the Windows [Services](#) console). We recommend that you do not use the `Local System` user, but a dedicated OS user for the `cloud connector` service.

- Only the file owner should have write permission for the file.
 - The OS user who runs the `cloud_connector` process needs read-only permissions for this file.
 - Initially, the file should contain a line like `allowed=false`.
 - In the security properties of the file `scc_config.ini` (same directory), make sure that only the OS user who runs the `cloud_connector` process has write/modify permissions for this file. The most efficient way to do this is simply by removing all other users from the list.
2. Once you've created this file, the Cloud Connector refuses any attempt to activate the *Payload Trace* or *SNC Payload Trace* flag.
 3. To set CPIC trace level 3 or activate any payload trace, first the owner of `writeHexDump` must change the file content from `allowed=false` to `allowed=true`. Thereafter, the *Administrator* user can activate any payload trace from the Cloud Connector administration screens.

Four-Eyes Principle for Linux OS/Mac OS X

1. Go to directory `/opt/sap/scc/scc_config` and create a file with name `writeHexDump`. The owner of this file must be different from the `scctunnel` user (that is, the operating system user under which the Cloud Connector processes are running) and not a member of the operating system user group `sccgroup`.
 - Only the file owner should have write permission for the file.
 - The `scctunnel` user needs read-only permissions for this file.
 - Initially, the file should contain a line like `allowed=false`.
2. Once you've created this file, the Cloud Connector refuses any attempt to activate the *Payload Trace* or *SNC Payload Trace* flag.
3. To set CPIC trace level 3 or activate any payload trace, first the owner of the `writeHexDump` file mentioned above must change the file content from `allowed=false` to `allowed=true`. Then, the *Administrator* user can activate any payload trace from the Cloud Connector administration screens.

1.2.3.8 Monitoring

Learn how to monitor the Cloud Connector from the SAP BTP cockpit and from the Cloud Connector administration UI.

Checking the Operational State

The simplest way to verify whether a Cloud Connector is running is to try to access its administration UI. If you can open the UI in a Web browser, the `cloud_connector` process is running.

- On Microsoft Windows operating systems, the `cloud_connector` process is registered as a Windows service, which is configured to start automatically after a new Cloud Connector installation. If the Cloud Connector server is rebooted, the `cloud_connector` process should also auto-restart immediately. You can check the state with the following command:

```
sc query "SAP Cloud Connector"
```

The line state shows the state of the service.

- On Linux operating systems, the Cloud Connector is registered as a daemon process and restarts automatically each time the `cloud_connector` process is down, for example, following a system restart. You can check the daemon state with the following command:

```
service scc_daemon status
```

To verify if a Cloud Connector is connected to a certain cloud subaccount, log on to the Cloud Connector administration UI and go to the [Subaccount Dashboard](#), where the connection state of the connected subaccounts is visible, as described in section [Connect and Disconnect a Cloud Subaccount \[page 658\]](#).

Monitoring from the Cockpit

The cockpit includes a [Connectivity](#) section, where users can check the status of the Cloud Connector(s) attached in the current subaccount, if any, as well as information about the Cloud Connector ID, version, used Java runtime, high availability setup (master and shadow instance), and so on (choose [Connectivity](#) [Cloud Connectors](#) [»](#)).

Access to this view is granted to:

- Neo environment: Users with a role containing the permission `readSCCTunnels`, for example, the predefined role `Cloud Connector Admin`.
- Cloud Foundry environment, feature set A: Users with a Cloud Foundry org role containing the permission `readSCCTunnels`, for example, the role `Org Manager`.

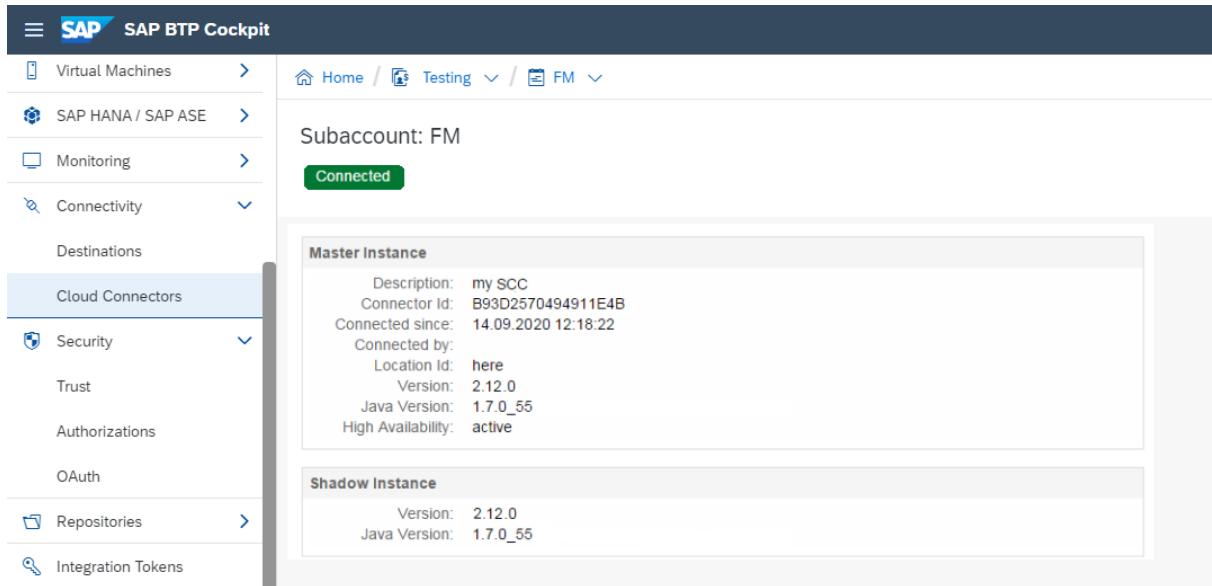
Note

As a prerequisite, a Cloud Foundry org must be available.

- Cloud Foundry environment, feature set B: Users with a role containing the permission `readSCCTunnels`, for example, the predefined role `Cloud Connector Administrator`.

Note

For more information on feature sets in the Cloud Foundry environment, see [Cloud Management Tools — Feature Set Overview](#).



Monitoring from the Cloud Connector Administration UI

The Cloud Connector offers various views for monitoring its activities and state.

You can check the overall state of the Cloud Connector through its [Hardware Metrics \[page 662\]](#), whereas subaccount-specific performance and usage data is available via [Subaccount-Specific Monitoring \[page 664\]](#). To provide external monitoring tools, you can use the [Monitoring APIs \[page 670\]](#).

Related Information

[Configure Solution Management Integration \[page 619\]](#)

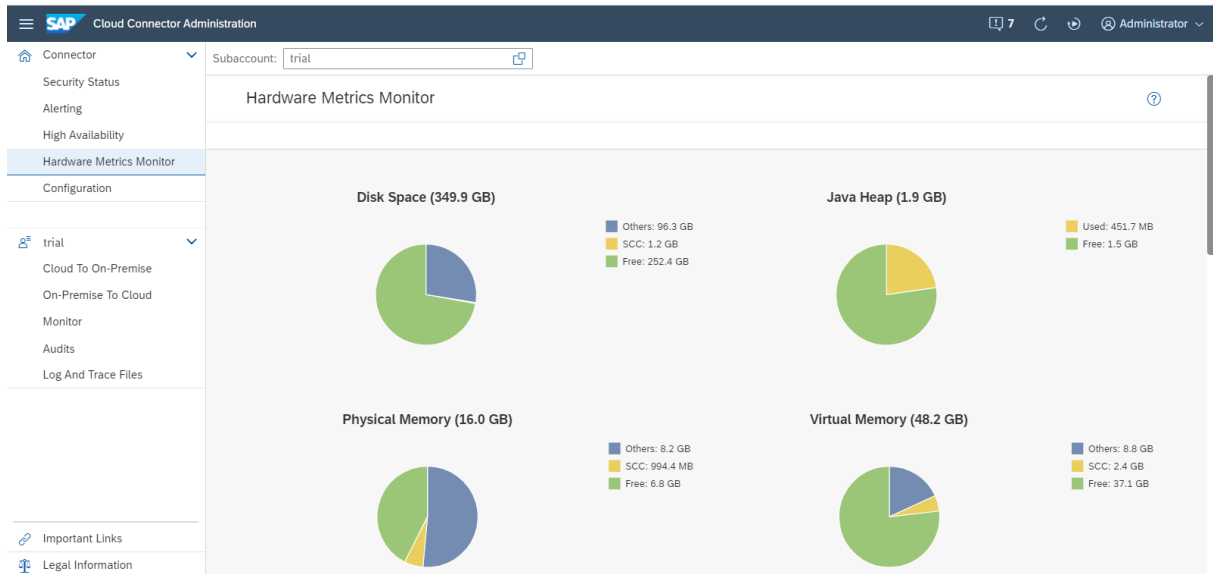
[High Availability Setup \[page 651\]](#)

1.2.3.8.1 Hardware Metrics

Check the current state of critical system resources in the Cloud Connector.

You can check the current state of critical system resources (disc space, Java heap, physical memory, virtual memory) using pie charts.

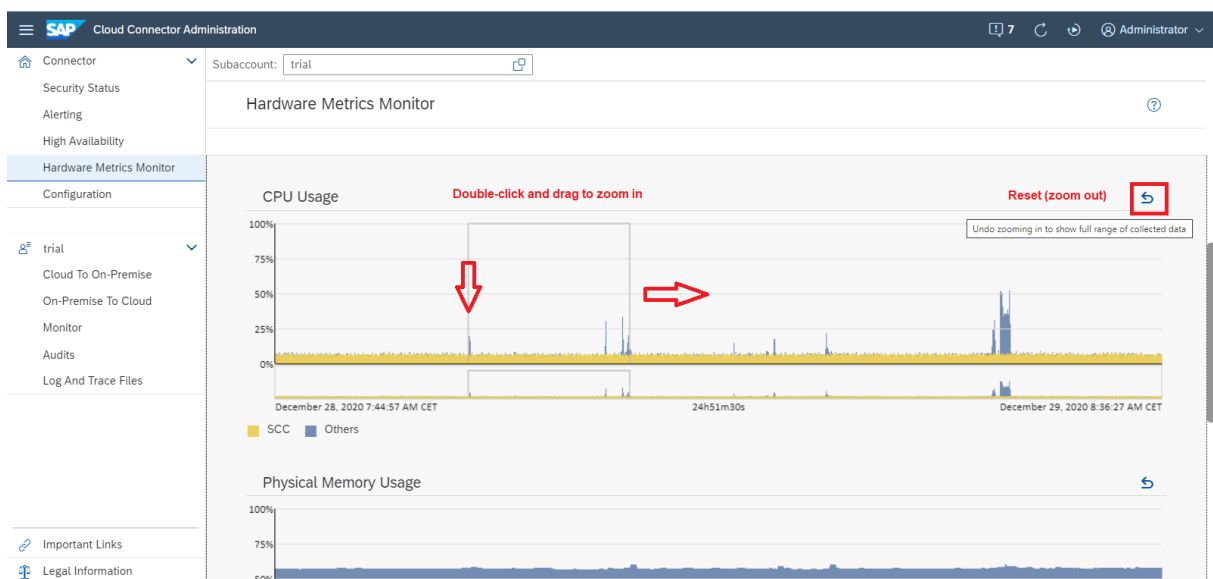
To access the monitor, choose [Hardware Metrics Monitor](#) from the main menu.



In addition, the history of CPU and memory usage (physical memory, Java heap) is shown in history graphs below the pie charts (recorded in intervals of 15 seconds), as well as the history of disk usage, recorded in intervals of 60 seconds. History data is retained for at most 24 hours.

You can view the usage data for a selected time period in each history graph:

- Double-click inside the main graph area to set the start (or end) point, and drag to the left or to the right to zoom in.
 - The entire timeline is always visible in the smaller bottom area right below the main graph.
 - A frame in the bottom area shows the position of the selected section in the overall timeline.
- Choose *Undo zooming in...* to reset the main graph area to the full range of available data.



Note

Zooming, dragging, and undoing zooming is synchronized across all history graphs. All graphs always show the situation during the same time period.

History data is written to a file to avoid data loss when stopping the Cloud Connector. Upon restart, the history data is read from file. Downtime is represented by a gray rectangle indicating that no data is available during the respective period of time.

1.2.3.8.2 Subaccount-Specific Monitoring

Use different monitoring views in the Cloud Connector administration UI to check subaccount-specific activities and data.

The Cloud Connector provides various views for monitoring the activities associated with an account, such as HTTP requests and RFC calls (from cloud applications to backends as per access control settings) or data statistics for service channels. Choose one of the sub-menus *Monitor (Cloud to On-Premise)* or *Monitor (On-Premise to Cloud)*.

Caution

The collected monitoring data is not part of the Cloud Connector's backup. After restoring a Cloud Connector instance, all monitoring collections are empty.

[Monitoring \(Cloud to On-Premise\) \[page 664\]](#)

[Monitoring \(On-Premise to Cloud\) \[page 669\]](#)

1.2.3.8.2.1 Monitoring (Cloud to On-Premise)

Monitor cloud to on-premise connections in the Cloud Connector.

Content

[Performance Overview \[page 665\]](#)

[Most Recent Requests \[page 665\]](#)

[Resource Filter Settings \[page 666\]](#)

[Top Time Consumers \[page 667\]](#)

[Usage Statistics \[page 668\]](#)

[Backend Connections \[page 669\]](#)

Performance Overview

All requests that travel through the Cloud Connector to a backend system, as specified through access control, take a certain amount of time. You can check the duration of requests in a bar chart. The requests are not shown individually, but are assigned to buckets, each of which represents a time range.

For example, the first bucket contains all requests that took 10ms or less, the second one the requests that took longer than 10ms, but not longer than 20ms. The last bucket contains all requests that took longer than 5000ms.

In case of latency gaps, you may try to adjust the influencing parameters: number of connections, tunnel worker threads, and protocol processor worker threads. For more information, see [Configuration Setup \[page 373\]](#).

The collection of duration statistics starts as soon as the Cloud Connector is operational. You can delete all of these statistical records by selecting the button *Delete All*. After that, the collection of duration statistics starts over.

Note

Delete All deletes not only the list of most recent requests, but it also clears the top time consumers.

Back to [Content \[page 664\]](#)

Most Recent Requests

This option shows the most recent requests:

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar contains navigation options like Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, trial, Cloud To On-Premise, On-Premise To Cloud, Monitor, Audits, and Log And Trace Files. The main area is titled 'Monitor' and shows performance statistics collected since 21. Dezember 2020 15:14:26 MEZ. Below this, there are tabs for PERFORMANCE, MOST RECENT REQUESTS (selected), TOP TIME CONSUMERS, USAGE STATISTICS, and BACK-END CONNECTIONS. Under 'Resource Filter Settings', there is a table with columns: Virtual Host, Exact Name or Path, Resource Filter, and Actions. The table contains one entry for 'sales-system.cloud:443'. Below this, there is a 'Select Host' dropdown menu set to 'All Hosts'. At the bottom, there is a table of requests with columns: Date, Virtual Host, Resource, Protocol, Duration, and Actions. Two requests are listed, both dated 'Dec 21, 2020 3:14:34 PM'. The first request has a duration of 68ms and the second has a duration of 42ms.

Date	Virtual Host	Resource	Protocol	Duration	Actions
Dec 21, 2020 3:14:34 PM	abapservers.hana.cloud:<port>	RFC_RAISE_ERROR	RFC	68ms	[Icon]
Dec 21, 2020 3:14:34 PM	abapservers.hana.cloud:<port>	RFCPING	RFC	42ms	[Icon]

The number of requests that are shown is limited to 50. You can either view all requests or only the ones destined for a certain virtual host, which you can select. You can select a row to see more detail.

Request Details

Basic Data

Date:	Dec 21, 2020 3:32:58 PM	Resource:	RFCPING
Protocol:	RFC	User:	JCOTEST
Virtual Host:	abapserver.hana.cloud:<port>	Bytes Sent:	304
Internal Host:	<host>:<port>	Bytes Received:	435

Duration Breakdown (34ms)



Close

A horizontal stacked bar chart breaks down the duration of the request into several parts: external (backend), open connection, internal (Cloud Connector), SSO handling, and latency effects (between SAP BTP and Cloud Connector). The numbers in each part represent milliseconds.

Note

Sections with a duration of less than 1ms are not included.

In the above example, the selected request took 34ms, to which the Cloud Connector contributed 1ms. Opening a connection took 18ms. Backend processing consumed 7ms. Latency effects accounted for the remaining 8ms, while there was no SSO handling necessary and hence it took no time at all.

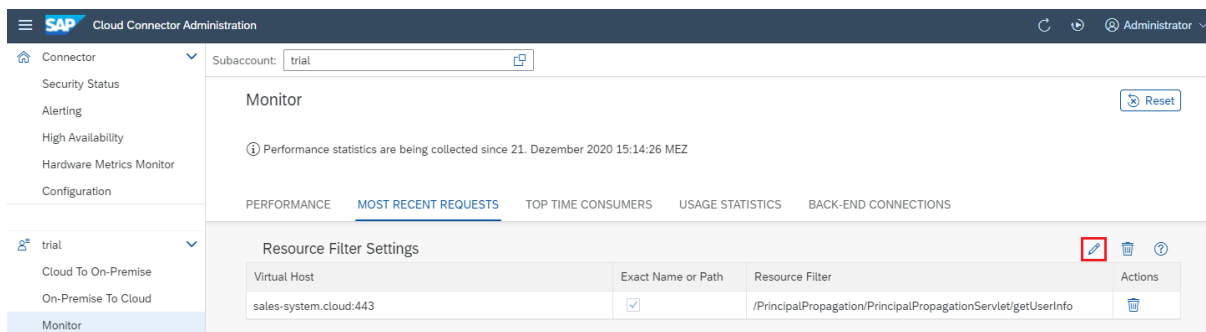
Note

The term "request" only refers to RFC and HTTP requests. TCP or LDAP traffic does not contribute to *Most Recent Requests* or *Top Time Consumers*.

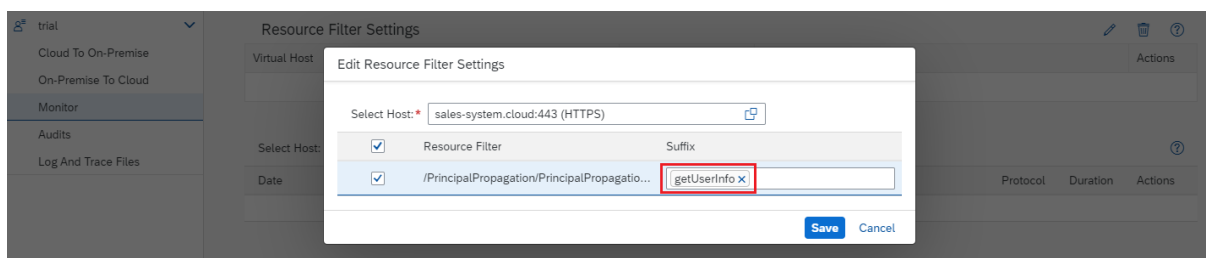
Back to [Content \[page 664\]](#)

Resource Filter Settings

To further restrict the selection of the listed 50 most recent requests, you can edit the resource filter settings for each virtual host:



In the *Edit* dialog, select the virtual host for which you want to specify the resource filter and choose one or more of the listed accessible resources. This list includes all resources that have been exposed during access control configuration (see also: [Configure Access Control \[page 456\]](#)). If the access policy for an accessible resource is set to `Path` and `all sub-paths`, you can further narrow the selection by adding one or more sub-paths to the resource as a suffix .



Each selected resource/sub-path is listed separately in the resource filter list.

Note

If you specify sub-paths for a resource, the request URL must match exactly one of these entries to be recorded. Without specified sub-paths (and the value `Path` and `all sub-paths` set for a resource), all sub-paths of a specified resource are recorded.

Back to [Content \[page 664\]](#)

Top Time Consumers

This option is similar to *Most Recent Requests*; however, requests are not shown in order of appearance, but rather sorted by their duration (in descending order). Furthermore, you can delete top time consumers, which has no effect on most recent requests or the performance overview.

Back to [Content \[page 664\]](#)

Usage Statistics

To view the statistical data regarding the traffic handled by each **virtual host**, you can select a virtual host from the table. The detail view shows the traffic handled by *each resource*, as well as a *24 hour overview* of the throughput as a bar chart that aggregates the throughput (bytes received and bytes sent by a virtual host, respectively) on an hourly basis.

Note

For communication via HTTP and RFC, also the number of calls or requests is recorded in the same way.

The screenshot displays the SAP Cloud Connector Administration interface. The left sidebar shows navigation options like Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, trial, Cloud To On-Premise, On-Premise To Cloud, Monitor, Audits, and Log And Trace Files. The main area is titled 'Monitor' and shows usage statistics for a subaccount named 'trial'. It includes tabs for PERFORMANCE, MOST RECENT REQUESTS, TOP TIME CONSUMERS, USAGE STATISTICS (selected), and BACK-END CONNECTIONS. Under 'Virtual Systems Usage (1)', a table lists a virtual host 'RFC' with 21.4 KB Bytes Received, 12.8 KB Bytes Sent, and 20 calls. Below this, 'Resources Usage Of' shows a resource 'STFC_CONNECTION' with 6190 Bytes Received and 4803 Bytes Sent. At the bottom, a '24H Throughput Overview' bar chart is partially visible, showing 'Bytes Received' on the y-axis.

The data that is collected includes the number of bytes received *from* cloud applications and the number of bytes sent back to cloud applications. The time of the most recent access is shown for the virtual hosts, and in the detail view also for the resources. If no access has taken place yet, the most recent access is shown as n . a . (not available). Similarly, the number of bytes received and sent of a virtual host is the sum of bytes received and sent of its resources.

The tables listing usage statistics of virtual hosts and their resources let you delete unused virtual hosts or unused resources. Use action *Delete* to delete such a virtual host or resource.

Caution

In Cloud Connector versions **before 2.14**, usage statistics are collected during runtime only and are not stored when stopping the Cloud Connector. That is, these statistics are lost when the Cloud Connector is stopped or restarted. Use care when taking the decision to delete a resource or virtual host based on its usage statistics.

As of version 2.14, usage statistics are periodically stored to disk. The collected statistics are still available after a restart of the Cloud Connector.

Actively deleted statistics are gone in either case.

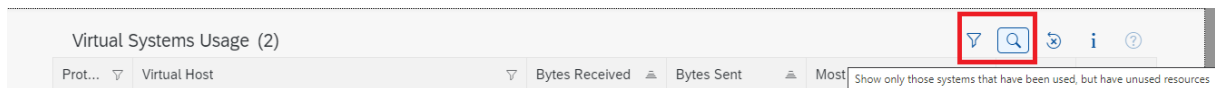
A backup of the Cloud Connector settings does not include collected usage data. After restoring a Cloud Connector instance, all monitoring collections are empty.

Using the [Reset](#) button, you can clean up all collected data.

⚠ Caution

Statistical data will be deleted permanently.

For both virtual hosts and resources, you can use a classic [Filter](#) button to reduce the virtual hosts or resources to those that have never been used (since the Cloud Connector started). For the virtual hosts, a second filter type is available that selects only those virtual hosts that have been used, but *include resources never used*. This feature facilitates locating obsolete resources of otherwise active virtual hosts.



Back to [Content \[page 664\]](#)

Backend Connections

This option shows a tabular overview of all active and idle connections, aggregated for each virtual host. By selecting a row (each of which represents a virtual host) you can view the details of all active connections as well as a graphical summary of all idle connections. The graphical summary is an accumulative view of connections based on the time the connections have been idle.

The maximum idle time appears on the rightmost side of the horizontal axis. For any point t on that axis (representing a time value ranging between 0ms and the maximal idle time), the ordinate is the number of connections that have been idle for no longer than t . You can click inside the graph area to view the respective abscissa t and ordinate.

Back to [Content \[page 664\]](#)

1.2.3.8.2.2 Monitoring (On-Premise to Cloud)

Monitor on-premise to cloud connections in the Cloud Connector.

Connections

This section shows a tabular overview of all currently opened logical connections, aggregated for each local port. You can identify if and how many connections are currently opened through this specific service channel.

Usage Statistics

Statistical data regarding the traffic handled by each port is shown in tabular form. From the table, you can select a service channel. The respective detail views show a 24 hour overview as a bar chart that aggregates the throughput (bytes received and bytes sent via a port, respectively) on an hourly basis.

The collected data comprises the number of bytes received from on-premise applications and the number of bytes sent to cloud applications.

The table listing usage statistics of ports lets you delete unused service channels. Use action [Delete](#) to delete a service channel.

⚠ Caution

Usage statistics are collected at runtime only. They are not stored when stopping the Cloud Connector. These statistics are lost when the Cloud Connector is stopped or restarted. Be mindful of that fact, and use caution when taking the decision to delete a service channel based on its usage statistics.

Using the [Reset](#) button you can clean up all collected data. This might be helpful when monitoring a specific use case.

The table of service channels provides a filter button to reduce the service channels to those that have never been used (since the Cloud Connector started).

1.2.3.8.3 Monitoring APIs

Use the Cloud Connector monitoring APIs to include monitoring information in your own monitoring tool.

Context

You might want to integrate some monitoring information in the monitoring tool you use.

For this purpose, the Cloud Connector includes a collection of APIs that allow you to read various types of monitoring data.

📌 Note

This API set is designed particularly for monitoring the Cloud Connector via the SAP Solution Manager, see [Configure Solution Management Integration \[page 619\]](#).

Before you start using these APIs, please also read the general introduction to [REST APIs \[page 728\]](#) provided by Cloud Connector.

Prerequisites

You must use *Basic Authentication* or *form field* authentication to read the monitoring data via API.

Users must be assigned to the roles `sccmonitoring` or `sccadmin`.

Note

The Health Check API does not require a specified user. Separate users are available through LDAP only.

Available APIs

The following APIs are currently available.

- [Health Check \[page 671\]](#) (available as of version 2.16.0)
- [Subaccount Data \[page 672\]](#) (as of 2.10.0)
- [Open Connections to On-Premise Backend Systems \[page 673\]](#) (as of 2.10.0)
- [Open Connections to Cloud Services \[page 674\]](#) (as of 2.15.0)
- [Performance Data \[page 675\]](#) (as of 2.10.0)
- [Top Time Consumers \[page 677\]](#) (as of 2.11.0)
- [Memory Status \[page 679\]](#) (as of 2.13.0)
- [Certificate Status \[page 681\]](#) (as of 2.13.0)
- [Certificate Selection List \[page 683\]](#) (as of 2.13.0)
- [Usage Statistics \[page 684\]](#) (as of 2.13.0)
- [Master Role Check \[page 687\]](#) (as of 2.15.0)

Health Check (available as of version 2.16.0)

Using the health check API, it is possible to recognize that the Cloud Connector is up and running. The purpose of this health check is only to verify that the Cloud Connector is not down. It does not check any internal state or tunnel connection states. Thus, it is a quick check that you can execute frequently:

URI	<code>/exposed?action=ping</code>
Method	<code>GET</code>
Request	
Response	
Errors	
Roles	All roles are accepted

Back to [Available APIs \[page 671\]](#)

Back to [Context \[page 670\]](#)

List of Subaccounts (available as of version 2.10.0)

Note

This API is relevant for the master instance only.

Using this API, you can read the list of all subaccounts connected to the Cloud Connector and view detail information for each subaccount:

URI	<code>/api/monitoring/subaccounts</code>
Method	GET
Request	
Response	
Errors	
Roles	Administrator, Monitoring

Response Properties:

- `subaccounts`: array of subaccounts for which data is provided
 - `regionHost`: host of the region, in which the subaccount is residing
 - `subaccount`: name of subaccount
 - `displayName`: display name of the subaccount
 - `description`: description for the subaccount
 - `locationID`: identifying the location of this Cloud Connector for a specific subaccount
 - `tunnel`: array of connection tunnels used by the subaccount
 - `state`: *Connected*, *ConnectFailure*, or *Disconnected*
 - `connectedSinceTimeStamp`: connection start time as UTC timestamp
 - `connections`: number of subaccount connections
 - `applicationConnections`: array of connections to application instances
 - `serviceChannels`: type and state of the service channels used (types: HANA database, Virtual Machine or RFC)
 - `subaccountCertificate`: information on the subaccount certificate such as validity period, issuer and subject DN
- `version`: API version.

Example:

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<scport>/api/monitoring/subaccounts
```

Back to [Available APIs \[page 671\]](#)

Back to [Context \[page 670\]](#)

List of Open Connections to On-Premise Backend Systems (available as of version 2.10.0)

Note

This API is relevant for the master instance only.

The list of connections lets you view all backend systems connected to the Cloud Connector and get detail information for each connection:

URI	<code>/api/monitoring/connections/backends</code>
Method	GET
Request	
Response	<code>{subaccounts, version}</code>
Errors	
Roles	Administrator, Monitoring

Response Properties:

- `subaccounts`: array of subaccounts for which data is provided
 - `regionHost`: host of the region, in which the subaccount is residing
 - `subaccount`: name of subaccount
 - `locationID`: identifying the location of this Cloud Connector for a specific subaccount
 - `backendConnections`: array of connections to a specified backend system
 - `virtualBackend`: virtual (external) backend URL
 - `internalBackend`: internal backend URL
 - `protocol`: type of protocol (RFC, HTTP, and so on)
 - `idle`: number of idle connections
 - `active`: number of active connections
- `version`: API version.

Sample Code

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<scport>/api/monitoring/connections/backends
```

Example:


```

{
  "subaccounts": [
    {
      "backendConnections": [],
      "regionHost": "hana.ondemand.com",
      "subaccount": "a117",
      "locationID": ""
    },
    {
      "backendConnections": [
        {
          "virtualBackend": "abapserver.hana.cloud:",
          "internalBackend": "sap.corp:",
          "protocol": "RFC",
          "idle": 1,
          "active": 0
        }
      ],
      "regionHost": "hana.ondemand.com",
      "subaccount": "dev",
      "locationID": ""
    },
    {
      "backendConnections": [],
      "regionHost": "hanatrial.ondemand.com",
      "subaccount": "trial",
      "locationID": ""
    }
  ],
  "version": 1
}

```

Back to [Available APIs \[page 671\]](#)

Back to [Context \[page 670\]](#)

List of Open Connections to Cloud Services (available as of version 2.15.0)

ⓘ Note

This API is relevant for the master instance only.

The list of connections opened for service channels:

URI	/api/monitoring/connections/serviceChannels
Method	GET

Request	
Response	{subaccounts, version}
Errors	
Roles	Administrator, Monitoring

Response Properties:

- **subaccounts**: array of subaccounts for which data is provided
 - **regionHost**: host of the region, in which the subaccount is residing
 - **subaccount**: name of subaccount
 - **locationID**: identifying the location of this Cloud Connector for a specific subaccount
 - **serviceChannelConnections**: array of connections opened by the specified service channel
 - **port**: port of the service channel
 - **typeDesc**:
 - **typeKey**: key name of the service channel, for example, *ABAPCloud*
 - **typeName**: human readable service channel type
 - **connections**: number of connections
- **version**: API version.

Sample Code

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<scport>/api/monitoring/connections/serviceChannels
```

Back to [Available APIs \[page 671\]](#)

Back to [Context \[page 670\]](#)

Performance Monitor Data (available as of version 2.10.0)

Note

This API is relevant for the master instance only.

Using this API, you can read the data provided by the Cloud Connector performance monitor:

URI	<code>/api/monitoring/performance/backends</code>
Method	GET
Request	

Response	{subaccounts, version}
Errors	
Roles	Administrator, Monitoring

Response Properties:

- **subaccounts:** array of subaccounts for which data is provided
 - **regionHost:** host of the region, in which the subaccount is residing
 - **subaccount:** name of subaccount
 - **locationID:** identifying the location of this Cloud Connector for a specific subaccount
 - **backendPerformance** given as array of:
 - **virtualHost:** host name of the backend system
 - **virtualPort:** port of the backend system
 - **protocol:** type of protocol (RFC, HTTP etc.)
 - **buckets:** array of performance data related to backend system
 - **numberOfCalls:** number of calls performed between Cloud Connector and backend system
 - **minimumCallDurationsMs:** minimum duration of the executed calls in milliseconds
 - **sinceTime:** start of performance measurement
- **version:** API version.

Sample Code

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<scport>/api/monitoring/performance/backends
```

Example:

```

{
  "subaccounts": [
    {
      "backendPerformance": [],
      "sinceTime": "2017-05-11T15:48:42.084 +0200",
      "regionHost": "hana.ondemand.com",
      "subaccount": "a117",
      "locationID": ""
    },
    {
      "backendPerformance": [
        {
          "virtualHost": "abapserver.hana.cloud",
          "virtualPort": "2",
          "protocol": "RFC",
          "buckets": [
            {
              "numberOfCalls": 0,
              "minimumCallDurationMs": 0
            },
            {
              "numberOfCalls": 1,
              "minimumCallDurationMs": 10
            },
            {
              "numberOfCalls": 0,
              "minimumCallDurationMs": 20
            },
            {
              "numberOfCalls": 0,
              "minimumCallDurationMs": 30
            }
          ]
        }
      ]
    }
  ]
}

```

Back to [Available APIs \[page 671\]](#)

Back to [Context \[page 670\]](#)

Top Time Consumers (available as of version 2.11.0)

📌 Note

This API is relevant for the master instance only.

Using this API, you can read the data of top time consumers provided by the Cloud Connector performance monitor:

URI	<code>/api/monitoring/performance/topTimeConsumers</code>
Method	GET
Request	
Response	<code>{subaccounts, version}</code>
Errors	
Roles	Administrator, Monitoring

Response Properties:

- `subaccounts`: array of subaccounts for which data is provided
 - `regionHost`: host of the region, in which the subaccount is residing
 - `subaccount`: name of subaccount
 - `locationID`: identifying the location of this Cloud Connector for a specific subaccount
 - `requests`: given as array of:
 - `protocol`: type of protocol (RFC, HTTP, and so on)
 - `virtualBackend`: virtual (external) backend URL
 - `internalBackend`: internal backend URL
 - `resource`: name of the request resource
 - `sentBytes`: number of sent bytes
 - `receivedBytes`: number of received bytes
 - `user`: name of the request user
 - `totalTime`: total request time in milliseconds
 - `externalTime`: in milliseconds
 - `genSsoTime`: in milliseconds
 - `openRemoteTime`: in milliseconds
 - `validationSsoTime`: time for SSO validation in milliseconds
 - `latencyTime`: latency in milliseconds
 - `sinceTime`: start of performance measurement
- `version`: API version.

Sample Code

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<scport>/api/monitoring/performance/topTimeConsumers
```

Example:

```

{
  "subaccounts": [
    {
      "sinceTime": "2017-06-26T16:57:21.615 +0200",
      "requests": [
        {
          "startTime": "2017-06-26T16:57:25.426 +0200",
          "id": 1639964397,
          "protocol": "RFC",
          "virtualBackend": "abapserver.hana.cloud:sapgw42",
          "internalBackend": "ldciv9u:sapgw51",
          "resource": "RFCPING",
          "sentBytes": 307,
          "receivedBytes": 441,
          "user": "JCOUSER",
          "totalTime": 21,
          "externalTime": 6,
          "genSsoTime": 0,
          "openRemoteTime": 5,
          "validateSsoTime": 0,
          "latencyTime": 6
        },
        ▶ { ... } // 15 items
      ],
      "regionHost": "int.sap.hana.ondemand.com",
      "subaccount": "d039407sapdev",
      "locationID": "location"
    }
  ],
  "version": 1
}

```

[Back to Available APIs \[page 671\]](#)

[Back to Context \[page 670\]](#)

Memory Status (available as of version 2.13.0)

📌 Note

This API is relevant for the master instance only.

This API provides a snapshot of the current memory status of the machine where the Cloud Connector is running:

URI	<code>/api/monitoring/memory</code>
Method	GET
Request	

Response`{physicalKB, virtualKB, cloudConnectorHeapKB}`**Errors****Roles**

Administrator, Monitoring

Response Properties:

- `physicalKB`: usage of the physical memory, split into four categories (all sizes in KB):
 - `total`: total size of the physical memory
 - `CloudConnector`: size of the physical memory used by the Cloud Connector
 - `others`: size of the physical memory used by all other processes
 - `free`: size of the free physical memory
- `virtualKB`: usage of the virtual memory, split into four categories (all sizes in KB)
 - `total`: total size of the virtual memory
 - `CloudConnector`: size of the virtual memory used by the Cloud Connector
 - `others`: size of the virtual memory used by all other processes
 - `free`: size of the free virtual memory
- `cloudConnectorHeapKB`: usage of the Java heap, split into three categories (all sizes in KB):
 - `total`: total size of the Java heap
 - `used`: size of the Java heap used by the Cloud Connector
 - `free`: size of the free Java heap

Sample Code

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://  
<scchost>:<scport>/api/monitoring/memory
```

Example:

```

{
  "physicalKB": {
    "total": 33380516,
    "CloudConnector": 633932,
    "others": 10960464,
    "free": 21786120
  },
  "virtualKB": {
    "total": 35477668,
    "CloudConnector": 1267504,
    "others": 14174208,
    "free": 20035956
  },
  "cloudConnectorHeapKB": {
    "total": 983040,
    "used": 235457,
    "free": 747583
  }
}

```

Back to [Available APIs \[page 671\]](#)

Back to [Context \[page 670\]](#)

Certificate Status (available as of version 2.13.0)

Note

This API is relevant for the master instance only.

Using this API, you can get an overview of the certificates currently employed by the Cloud Connector:

URI	<code>/api/monitoring/certificates</code>
Method	GET
Request	
Response	<code>{expired, expiring, ok}</code>
Errors	
Roles	Administrator, Monitoring

Response Properties:

- `expired`: list of all expired certificates
- `expiring`: list of all certificates that will expire in less than N days, where N is the number of days specified in the alerting setup regarding certificates that are close to their expiration date

- `ok`: list of all certificates that continue to be valid for N days or more, where N is the number of days specified in the alerting setup regarding certificates that are close to their expiration date

A certificate in any of those lists is represented by a JSON object with the following properties:

- `type`: type of the certificate which can be one of the following strings:
 - `UI` (for the UI certificate)
 - `System` (for the system certificate)
 - `CA` (for the certificate used in connection with Principal Propagation/Certification Authority)
 - `subaccount` (for subaccount certificates)
- `validTo`: end date of the respective certificate's validity (as a long integer, that is, a UTC timestamp)
- `subjectDN`: subject DN of the respective certificate (included only for non-subaccount certificates)
- `issuerDN`: issuer DN of the respective certificate (included only for non-subaccount certificates)
- `serialNumber`: serial number of the respective certificate as hex-encoded string (included only for non-subaccount certificates)
- `subaccountName`: name of the subaccount (only for subaccount certificates)
- `subaccountRegion`: region or landscape host of the the subaccount (only for subaccount certificates)

Sample Code

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<scport>/api/monitoring/certificates
```

Example:

```
{
  "expired": [
    {
      "type": "System",
      "subjectDN": "CN\u003dHugo, OU\u003dCSI, O\u003dSAP Trust Community, C\u003dDE",
      "validTo": 1458290342000
    }
  ],
  "expiring": [],
  "ok": [
    {
      "type": "UI",
      "subjectDN": "CN\u003dSCC, OU\u003dConnectivity, O\u003dSAP SE, C\u003dDE",
      "validTo": 1791826434000
    },
    {
      "type": "subaccount",
      "subaccountName": "d036325trial",
      "subaccountRegion": "hanatrial.ondemand.com",
      "validTo": 1613132865000
    }
  ]
}
```

Back to [Available APIs \[page 671\]](#)

Back to [Context \[page 670\]](#)

Certificate Selection List (available as of version 2.13.0)

Note

This API is relevant for the master instance only.

Using this API, you can obtain an overview of the certificates currently employed by the Cloud Connector:

URI	<code>/api/monitoring/certificates/{selection}</code>
Method	GET
Request	
Response	<code>[Array of certificates]</code>
Errors	
Roles	Administrator, Monitoring

Request:

- `selection` parameter
 - `expired`: an array holding the list of all expired certificates
 - `expiring`: an array holding the list of all certificates that will expire in less than N days, where N is the number of days specified in the alerting setup regarding certificates that are close to their expiration date
 - `ok`: an array holding the list of all certificates that continue to be valid for N days or more, where N is the number of days specified in the alerting setup regarding certificates that are close to their expiration date

Response Properties:

- `type`: type of the certificate which can be one of the following strings:
 - `UI` (for the UI certificate)
 - `System` (for the system certificate)
 - `CA` (for the certificate used in connection with Principal Propagation/Certification Authority)
 - `subaccount` (for subaccount certificates)
- `validTo`: end date of the respective certificate's validity (as a long integer, that is, a UTC timestamp)
- `subjectDN`: subject DN of the respective certificate (included only for non-subaccount certificates)
- `subaccountName`: name of the subaccount (only for subaccount certificates)
- `subaccountRegion`: region or landscape host of the the subaccount (only for subaccount certificates)

Sample Code

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<scport>/api/monitoring/certificates/expired
```

Back to [Available APIs \[page 671\]](#)

Back to [Context \[page 670\]](#)

Usage Statistics (available as of version 2.13.0)

Note

This API is relevant for the master instance only.

This API provides usage statistics regarding the systems and resources available in the Cloud Connector:

URI	<code>/api/monitoring/usage</code>
Method	GET
Request	
Response	<code>{subaccounts, version}</code>
Errors	
Roles	Administrator, Monitoring

Response Properties:

- `subaccounts`: array of subaccounts for which data is provided
 - `regionHost`: host of the region, in which the subaccount is residing
 - `subaccount`: name of subaccount
 - `locationID`: identifying the location of this Cloud Connector for a specific subaccount
 - `usageStatistics`: backend usage statistics, given as an array of
 - `virtualHost`: host name of the backend system
 - `virtualPort`: port of the backend system
 - `protocol`: type of protocol (RFC, HTTP etc.)
 - `bytesReceived`: total number of bytes that were received through a call or request
 - `bytesSent`: total number of bytes sent back as a response
 - `calls`: total number of calls or requests
 - `mostRecentAccess`: time of the most recent access (that is, call or request) given as a UTC timestamp;

Note

This property is only available if there has been at least one call or request.

- **resources**: usage statistics per resource, given as an array (i.e. the distribution of bytes received, sent, as well as number of calls/requests, across the resources of the respective virtual host)
 - **resourceName**: name of the resource (that is, a URL path or the name of a remote function)
 - **enabled**: Boolean flag that indicates whether the resource is currently active (true) or suspended (false)
 - **bytesReceived**: total number of bytes that were received through a call or request and were handled by this resource
 - **bytesSent**: total number of bytes sent back as a response in the context of this resource
 - **calls**: total number of calls or requests handled by this resource
 - **mostRecentAccess**: time of the most recent access (that is, call or request) given as a UTC timestamp;

Note

This property is only available if at least one call or request was handled by this resource.

- **serviceChannelUsageStatistics**: service channels usage statistics, given as an array of
 - **port**: port of the service channel
 - **typeDesc**:
 - **typeKey**: key name of the service channel, for example, *ABAPCloud*
 - **typeName**: human readable service channel type
 - **bytesReceived**: total number of bytes received through the service channel
 - **bytesSent**: total number of bytes sent back through the service channel
- **sinceTime**: start of performance measurement
- **version**: API version.

Sample Code

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<scport>/api/monitoring/usage
```

Example:

```

{
  "subaccounts": [
    {
      "subaccountName": "lannister",
      "subaccountRegion": "westeros.com",
      "sinceTime": 1596810677689,
      "usageStatistics": [
        {
          "virtualHost": "iron.islands",
          "virtualPort": "1234",
          "protocol": "RFC",
          "bytesReceived": 0,
          "bytesSent": 0,
          "calls": 0,
          "resources": [
            {
              "resourceName": "TEST_FCT_1",
              "enabled": true,
              "bytesReceived": 0,
              "bytesSent": 0,
              "calls": 0
            },
            {
              "resourceName": "TEST_FCT_2",
              "enabled": true,
              "bytesReceived": 0,
              "bytesSent": 0,
              "calls": 0
            }
          ]
        },
        {
          "virtualHost": "the.north",
          "virtualPort": "8080",
          "protocol": "HTTP",
          "bytesReceived": 1482,
          "bytesSent": 705,
          "calls": 3,
          "mostRecentAccess": 1596810970078,
          "resources": [
            {
              "resourceName": "/invoke/giants",
              "enabled": true,
              "bytesReceived": 0,
              "bytesSent": 0,
              "calls": 0
            },
            {
              "resourceName": "/invoke/whitewalkers",
              "enabled": true,
              "bytesReceived": 0,
              "bytesSent": 0,
              "calls": 0
            },
            {
              "resourceName": "/invoke/nightwatch",
              "enabled": true,
              "bytesReceived": 1482,
              "bytesSent": 705,
              "calls": 3,
              "mostRecentAccess": 1596810970078
            }
          ]
        }
      ]
    }
  ]
}

```

Back to [Available APIs \[page 671\]](#)

Back to [Context \[page 670\]](#)

Master Role Check (available as of version 2.15.0)

With the master role check API, you can recognize if a Cloud Connector instance has currently the master role. The purpose of this master role check is only to recognize if the Cloud Connector instance is currently the master instance or not, without the need of providing credentials. It is a quick check that you can execute frequently.

URI	<code>/exposed?action=hasMasterRole</code>
Method	GET
Request	
Response	<code>{true, false}</code>
Errors	
Roles	All roles are accepted

Back to [Available APIs \[page 671\]](#)

Back to [Context \[page 670\]](#)

1.2.3.9 Alerting

Configure the Cloud Connector to send e-mail messages when situations occur that may prevent it from operating correctly.

To configure alert e-mails, choose *Alerting* from the top-left navigation menu.

You must specify the receivers of the alert e-mails (*E-mail Configuration*) as well as the Cloud Connector resources and components that you want to monitor (*Observation Configuration*). The corresponding *Alert Messages* are also shown in the Cloud Connector administration UI.

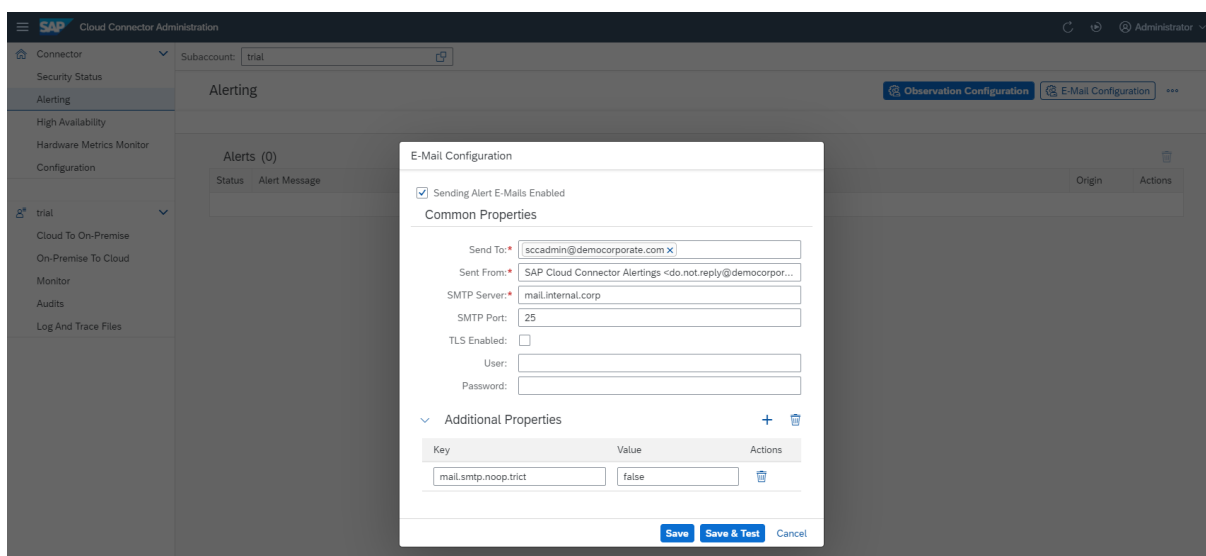
E-mail Configuration

1. Select *E-mail Configuration* to specify the list of em-ail addresses to which alerts should be sent (*Send To*).

Note

The addresses you enter here can use either of the following formats: john.doe@company.com or John Doe <j.doe@company.com>.

2. Enter the sender's e-mail address (<Sent From>).
3. In <SMTP Server> provide the host of the mail server.
4. You can specify an <SMTP port>, if the server is not using the default ports. For details, contact your e-mail administrator or provider.
5. Mark the *TLS Enabled* check box if you want to establish a TLS-encrypted connection.
6. If the SMTP server requires authentication, provide <User> and <Password>.
7. In the *Additional Properties* section you can provide any property supported by the [Java Mail library](#) . All specified properties will be passed to the SMTP client.
8. Select *Save* to change the current configuration.



Note

Connections to an SMTP server over TLS can cause TLS errors if the SMTP server uses an "untrusted" certificate. If you cannot use a trusted certificate, you must import the public part of the issuer certificate to the JDK's trust storage.

Usually, the trust storage is done in the file *cacerts* in the Java directory (*jre/lib/security/cacerts*). For import, you can use the *keytool* utility:

```
keytool -import -storepass changeit -file <certificate used by SMTP server>
-keystore cacerts -alias <for example, SMTP_xyz>
```

For more information, see <https://docs.oracle.com/cd/E19830-01/819-4712/ablqw/index.html> .

Observation Configuration

Once you've entered the e-mail addresses to receive alerts, the next step is to identify the resources and components of the Cloud Connector: E-mail messages are sent when any of the chosen components or resources have malfunctioned or are in a critical state.

Note

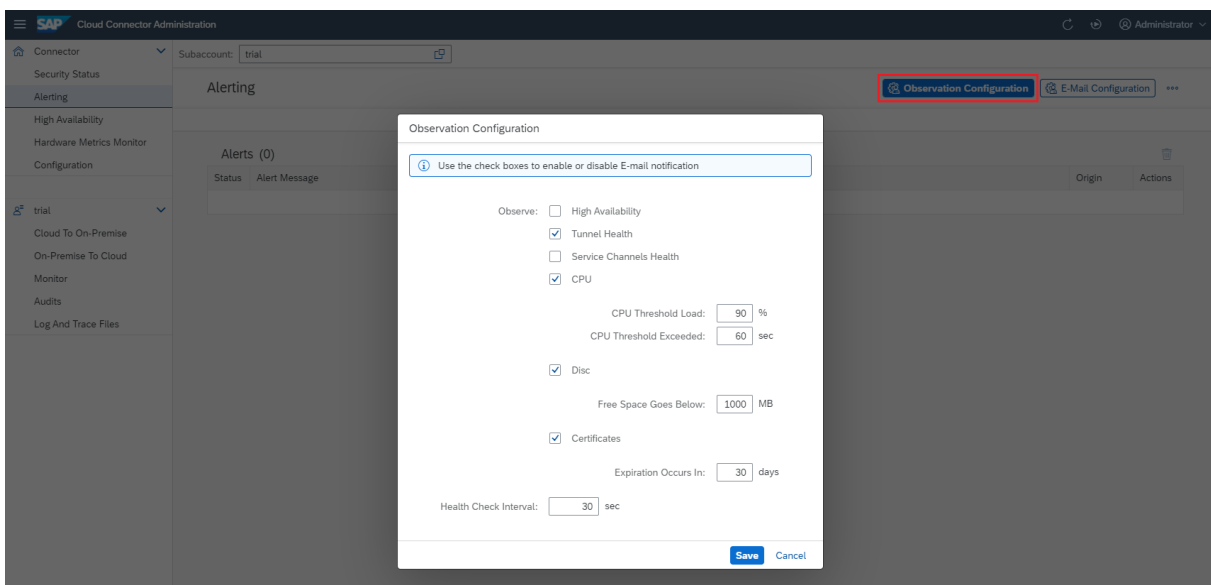
The Cloud Connector does not dispatch the same alert repeatedly. As soon as an issue has been resolved, an informational alert is generated, sent, and listed in [Alert Messages](#) (see section below).

1. Select [Observation Configuration](#) from the top-right corner of the window.
2. Select the components or resources you want to monitor.
 - [High Availability](#) alerts can occur in the context of an active high availability setup, meaning a shadow system is connected.
 - [Tunnel Health](#) and [Service Channels Health](#) refer to the state of the respective connections. Whenever such a connection is lost, an alert is triggered.

Note

These alerts are only triggered in case of an error or exception, but not upon intentional disconnect action.

- An excessively high [CPU](#) load over an extended period of time adversely affects performance and may be an indicator of serious issues that jeopardize the operability of the Cloud Connector. The CPU load is monitored and an alert is triggered whenever the CPU load exceeds and continues to exceed a given threshold percentage (the default is 90%) for more than a given period of time (the default is 60 seconds).
 - Although the Cloud Connector does not require nor consume large amounts of [Disk](#) space, running out of it is a circumstance that you should avoid. We recommend that you configure an alert to be sent if the disk space falls below a critical value (the default is 10 megabytes).
 - The Cloud Connector configuration contains various [Certificates](#). Whenever one of those expires, scenarios might no longer work as expected so it's important to get notified about the expiration (the default is 30 days).
3. (Optional) Change the [Health Check Interval](#) (the default is 30 seconds).
 4. Select [Save](#) to change the current configuration.



Alert Messages

The Cloud Connector shows alert messages also on screen, in [Alerting > Alert Messages](#).

You can remove alerts using [Delete](#) or [Delete All](#). If you delete active (unresolved) alerts, they reappear in the list after the next health check interval.

1.2.3.10 Audit Logging

Audit log data can alert Cloud Connector administrators to unusual or suspicious network and system behavior.

Additionally, the audit log data can provide auditors with information required to validate security policy enforcement and proper segregation of duties. IT staff can use the audit log data for root-cause analysis following a security incident.

The Cloud Connector includes an auditor tool for viewing and managing audit log information about access between the cloud and the Cloud Connector, as well as for tracking of configuration changes done in the Cloud Connector. The written audit log files are digitally signed by the Cloud Connector so that their integrity can be checked, see [Manage Audit Logs \[page 690\]](#).

Note

We recommend that you permanently switch on Cloud Connector audit logging in productive scenarios.

- Under normal circumstances, set the logging level to **Security** (the default configuration value).
- If legal requirements or company policies dictate it, set the logging level to **All**. This lets you use the log files to, for example, detect attacks of a malicious cloud application that tries to access on-premise services without permission, or in a forensic analysis of a security incident.

We also recommend that you regularly copy the audit log files of the Cloud Connector to an external persistent storage according to your local regulations. The audit log files can be found in the Cloud Connector root directory `/log/audit/<subaccount-name>/audit-log_<timestamp>.csv`.

1.2.3.10.1 Manage Audit Logs

Configure audit log settings and verify the integrity of audit logs.

Configure Audit Logs in the Cloud Connector

Choose [Audit](#) from your subaccount menu and go to [Settings](#) to specify the type of audit events the Cloud Connector should log at runtime. You can currently select between the following [Audit Levels](#) (for either `<subaccount>` and `<cross-subaccount>` scope):

- **Security:** Default value. The Cloud Connector writes an audit entry (`Access Denied`) for each request that was blocked. It also writes audit entries, whenever an administrator changes one of the critical configuration settings, such as exposed back-end systems, allowed resources, and so on.
- **All:** The Cloud Connector writes one audit entry for each received request, regardless of whether it was allowed to pass or not (`Access Allowed` and `Access Denied`). It also writes audit entries that are relevant to the **Security** mode.
- **Off:** No audit entries are written.

⚠ Caution

To prevent a single person from being able to both change the audit log level, and delete audit logs, we recommend that the operating system administrator and the SAP BTP administrator are different persons. We also suggest that you turn on the audit log at the operating system level for file operations.

→ Tip

We recommend that you don't log all events unless you are required to do so by legal requirements or company policies. Generally, logging security events only is sufficient.

To enable automatic cleanup of audit log files, choose a period (14 to 365 days) from the list in the field `<Automatic Cleanup>`.

Audit entries for configuration changes are written for the following categories:

- `Account`: A subaccount configuration was changed.
- `Configuration`: A new subaccount was added or a disaster recovery switch happened.
- `BackendMapping`: Changes to the virtual to internal system mappings.
- `AllowedResource`: In a virtual system, changes in the accessible resources.
- `DomainMapping`: Changes to the domain mappings.
- `ServiceChannelConfiguration`: The configuration of a service channel was changed.
- `SCCPassword`: The Cloud Connector administration password was changed.
- `SCCUser`: The Cloud Connector administration user was changed.
- `LDAPConfiguration`: Changes to the LDAP settings.
- `EMailConfiguration`: The Email settings for alerts were changed.
- `AlertConfiguration`: The observation settings for alerts were changed.
- `ScimConfiguration`: Something changed in the settings for the cloud user store.
- `KerberosConfiguration`: The Kerberos configuration was changed.
- `SNCSettings`: SNC settings of Cloud Connector were changed.
- `ProxySettings`: The proxy settings were changed.
- `SystemCertificate`: The system certificate was changed.
- `PpcaCertificate`: The CA certificate was changed.
- `PrincipalPropagationConfiguration`: The principal propagation settings were changed.
- `TrustSynchronization`: The trust configuration for principal propagation was synchronized.
- `IdentityProviderTrust`: The trust configuration for a specific identity provider was changed.
- `ApplicationTrust`: The trust configuration to applications was changed.
- `TrustedBackendCertificate`: The trust store certificate was added or removed.

- `SecCustomRoles`: Custom role name settings were changed.
- `BackendAuthority`: RFC-specific user and client settings were adjusted.
- `AdvancedConnectivity`: Advanced connectivity configuration was changed.
- `AdvancedJVM`: Advanced JVM configuration was changed.
- `ApplicationConfiguration`: Application-specific connection configuration was changed.
- `PayloadTrace`: Payload trace (traffic data) was activated/deactivated.
- `CPICTrace`: The CPIC trace level was changed.
- `AuditLogLevel`: The subaccount-specific audit log level was changed.
- `CrossAuditLogLevel`: The cross-subaccount audit log level was changed.
- `AuditLogCleanup`: The audit log cleanup setting was changed.

In the [Audit Viewer](#) section, you can first define filter criteria, then display or download the selected audit entries.

- In the [Audit Type](#) field, you can select the types of auditing events you are interested in.
- In the [Pattern](#) field, you can specify a certain string that the detail text of each selected audit entry must contain. The detail text may contain, for example, information about the user name, requested resource/URL, or the virtual `<host>: <port>`. Basic wildcards (glob patterns) are supported, that is, asterisk for any number of characters (including none) and question mark for a single character. Use this feature to do the following:
 - Filter the audit log for all requests that a particular HTTP user has made during a certain time frame.
 - Identify all users who attempted to request a particular URL.
 - Identify all requests to a particular back-end system.
 - Determine whether a user has changed a certain Cloud Connector configuration. For example, a search for string `BackendMapping` returns all `add-`, `delete-` or `modify-` operations on the [Mapping Virtual To Internal System](#) page.
- The [Time Range](#) settings specify the time frame to which you want to limit the eligible audit entries.

These filter criteria are combined with a logical **AND** so that all audit entries that match these criteria are shown. If you have modified the criteria, choose the [Search](#) button again to display the updated selection of audit events that match the new criteria.

Use the [Download](#) button to download the selected audit entries as a compressed (GZIP) CSV file that can be imported, for example, by Excel. The ZIP file also contains a second text file, a manifest or information, that provides the selection parameters as well as the date and time when the selection was extracted.

Example

In the following example, the [Audit Viewer](#) displays **ANY** audit entries, at **Security** level, for the time frame between **December 18 2020, 00:00:00** and **December 19, 00:00:00**:

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar contains navigation options: Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, trial, Cloud To On-Premise, On-Premise To Cloud, Monitor, Audits, and Log And Trace Files. The main area displays the 'Audits' section for subaccount 'trial'. It includes a 'Settings' section with the following values: Subaccount Audit Level: Security, Cross-Subaccount Audit Level: Security, and Automatic Cleanup: Never. Below the settings is a filter bar with the following fields: Audit Type: Any, Search Pattern: (empty), Start Time: 2020-12-18 00:00:00, and End Time: 2020-12-19 00:00:00. A table of audit logs is displayed below the filter bar:

Timestamp	Audit Log
2020-12-18 14:11:21 +0100	Audit Service is started for _crossaccount
2020-12-18 14:11:27 +0100	Audit Service is started for@hana.ondemand.com
2020-12-18 14:11:29 +0100	User during startup started Service Tunnel/account:///

Verify the Integrity of Audit Logs

To check the integrity of all or a part of the audit logs, go to `<scs_installation>/auditor`. This directory contains an executable `go` script file (respectively, `go.cmd` on Microsoft Windows and `go.sh` on other operating systems).

If you start the `go` file without specifying parameters from `<scs_installation>/auditor`, all available audit logs for the current Cloud Connector installation are verified.

The auditor tool is a Java application, and therefore requires a Java runtime, specified in `JAVA_HOME`, to execute:

- For Microsoft Windows OS, set `JAVA_HOME=<path-to-java-installation>`
- For Linux OS and Mac OS X, export: `JAVA_HOME=<path-to-java-installation>`

Alternatively, to execute Java, you can include the Java `bin` directory in the `PATH` variable.

You can check audit logs also in the UI, using the rightmost button of the **Audits** section.

Note

The selected date range determines the audit logs that are going to be checked (entire days only, ignoring the time of day).

Change the Location of Audit Logs

As of Cloud Connector 2.14 you can move audit logs to a different location. Standard location remains `log/audit`.

Note

Make sure there is enough space left on the device for the desired location and the Cloud Connector OS user has permission to write files to that location.

If you want to do this, proceed as follows:

1. Shut down the Cloud Connector.
2. Execute the respective script for the location change.
 1. For Microsoft Windows OS: `changeAuditLogPath.bat <desiredLocation>`.
 2. For Linux OS and Mac OS X: `./changeAuditLogPath.sh <desiredLocation>`.
3. The script checks if the target might be a network location. If this is assumed, the script asks for confirmation. Afterwards, it tries to move all existing audit logs to the new location. Only after successful move, the location change takes effect, otherwise it remains in the old place.
4. Start the Cloud Connector again.

Caution

If you choose a network location while access to the file system is slow, overall processing performance of the Cloud Connector may decrease significantly.

1.2.3.11 Troubleshooting

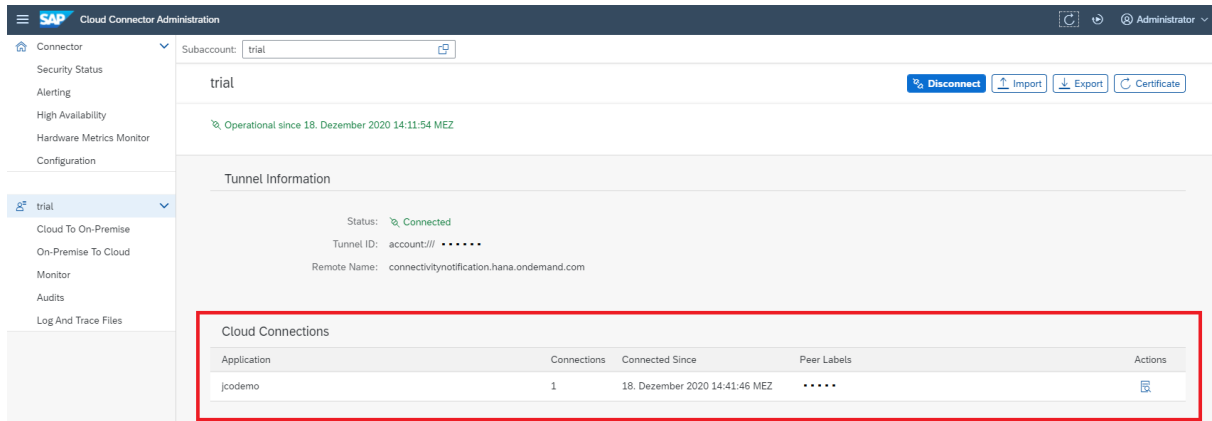
To troubleshoot connection problems, monitor the state of your open tunnel connections in the Cloud Connector, and view different types of logs and traces.

Note

For information about a specific problem or an error you have encountered, see [Connectivity Support \[page 869\]](#).

Monitoring

To view a list of all currently connected applications, choose your *Subaccount* from the left menu and go to section *Cloud Connections*:



The provided information includes:

- **Application name:** The name of the application, as also shown in the cockpit, for your subaccount
- **Connections:** The number of currently existing connections to the application
- **Connected Since:** The earliest start time of a connection to this application
- **Peer Labels:** The name of the application processes, as also shown for this application in the cockpit, for your subaccount

Log and Trace Settings

The [Log and Trace Files](#) page includes some files for troubleshooting that are intended primarily for SAP Support. These files include information about both internal Cloud Connector operations and details about the communication between the local and the remote (SAP BTP) tunnel endpoint.

If you encounter problems that seem to be caused by some trouble in the communication between your cloud application and the on-premise system, choose [Log and Trace Files](#) from your [Subaccount](#) menu, go to section [Settings](#), and activate the respective traces by selecting the [Edit](#) button:

- **Cloud Connector Loggers** adjusts the levels for Java loggers directly related to Cloud Connector functionality.
- **Other Loggers** adjusts the log level for all other Java loggers available at the runtime. Change this level only when requested to do so by SAP support. When set to a level higher than `Information`, it generates a large number of trace entries.
- **CPIC Trace Level** allows you to set the level between 0 and 3 and provides traces for the CPIC-based RFC communication with ABAP systems.
- When the **Payload Trace** is activated for a subaccount, all the HTTP and RFC traffic crossing the tunnel for that subaccount going through this Cloud Connector, is traced in files with names `traffic_trace_<subaccount id>_on_<regionhost>.trc`. This is helpful if you need to understand what documents have been exchanged between the involved systems.
- **Payload SNC Trace:** When the **Payload SNC trace** is activated for an account, all RFC SNC-based traffic crossing a service channel for that account (going through this Cloud Connector), is traced in files with names `payload_snc_trace_<account id>_on_<landscapehost>.trc`. This is helpful if you need to understand issues with SNC termination in the Cloud Connector.

- **SSL Trace:** When the SSL trace is activated, the `1js_trace.log` file includes information for SSL-protected communication. To activate a change of this setting, a restart is required. Activate this trace only when requested by SAP support. It has a high impact on performance as it produces a large amount of traces.
- **Automatic Cleanup** lets you remove old trace files that have not been changed for a period of time exceeding the configured interval. You can choose from a list of predefined periods. The default is `Never`.

Edit Log Settings

Cloud Connector Loggers:	Information	▼
Other Loggers:	Information	▼
CPIC Trace Level:	0	▼
Payload Trace:	<input type="checkbox"/>	
SSL Trace:	<input type="checkbox"/>	
Automatic Cleanup:	After 365 Days	▼

Save

Cancel

⚠ Caution

Use any **payload and CPIC tracing at level 3** carefully, and only when requested to do so for support reasons. These traces may write sensitive information (such as payload data of HTTP/RFC requests and responses) to the trace files, and thus present a potential security risk. The Cloud Connector supports the implementation of a "four-eyes principle" for activating the trace levels that dump the network traffic into a trace file. This principle requires two users to activate a trace level that records traffic data.

For more information, see [Secure the Activation of Traffic Traces \[page 659\]](#).

Change the Location of Trace Files

As of Cloud Connector 2.14 you can move trace files to a different location.

📌 Note

JVM-related files will remain in the standard location `log`.

Note

Make sure there is enough space left on the device for the desired location and the Cloud Connector OS user has permission to write files to that location.

If you want to do this, proceed as follows:

1. Shut down the Cloud Connector.
2. Execute the respective script for the location change.
 1. For Microsoft Windows OS: `changeLogAndTracePath.bat <desiredLocation>`.
 2. For Linux OS and Mac OS X: `./changeLogAndTracePath.sh <desiredLocation>`.
3. The script checks if the target might be a network location. If this is assumed, the script asks for confirmation. Afterwards, it tries to move the existing current trace file to the new location. Only after successful move, the location change takes effect. Otherwise, the file remains in the old place.
4. Start the Cloud Connector again.

Caution

If you choose a network location while access to the file system is slow, overall processing performance of the Cloud Connector may decrease significantly.

Log and Trace Files

View all existing trace files and delete the ones that are no longer needed.

The screenshot displays the SAP Cloud Connector Administration web interface. The left sidebar contains navigation links for 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. Under 'Connector', there is a sub-menu for 'trial' with options like 'Cloud To On-Premise', 'On-Premise To Cloud', 'Monitor', 'Audits', and 'Log And Trace Files'. The main content area is titled 'Log And Trace Files' and includes buttons for 'Thread Dump', 'Guided Answers', and 'Support Log Assistant'. Below this is a 'Settings' section with a list of loggers and their levels: Cloud Connector Loggers (Information), Other Loggers (Information), CPIC Trace Level (0), Payload Trace (No), SSL Trace (No), and Automatic Cleanup (Never). The 'Log And Trace Files (4)' table is as follows:

Last Modified	File Name	File Size	Actions
2020-12-18 14:41:53 +0100	localhost_http_access_2020-12-18.log	16.8 KB	Download Refresh Delete
2020-12-18 14:41:46 +0100	ljs_trace.log	59.8 KB	Download Refresh Delete
2020-12-18 14:11:00 +0100	vm_31400_gc.prf	0 Bytes	Download Refresh Delete
2020-12-17 15:51:04 +0100	localhost_http_access_2020-12-17.log	23.7 KB	Download Refresh Delete

At the bottom, the 'LJS Log Summary' bar shows: All 10 12 175.

To prevent your browser from being overloaded when multiple large files are loaded simultaneously, the Cloud Connector loads only one page into memory. Use the page buttons to move through the pages.

Use the [Download/Download All](#) icons to create a ZIP archive containing one trace file or all trace files. Download it to your local file system for convenient analysis.

Note

If you want to download more than one file, but not all, select the respective rows of the table and choose [Download All](#).

When running the Cloud Connector with SAP JVM or as of version 2.14 also with other JVMs, you can trigger the creation of a thread dump by choosing the [Thread Dump](#) button, which will be written to the JVM trace file `log/vm_${PID}_trace.log` for SAP JVM and `log/vm_${PID}_threads.log` for other JVMs. You may be asked by SAP support to create one, if considered helpful during incident analysis.

Note

From the UI, you can't delete trace files that are currently in use. You can delete them from the Linux OS command line; however, we recommend that you do not use this option to avoid inconsistencies in the internal trace management of the Cloud Connector.

Two buttons may be helpful to solve issues on your own:

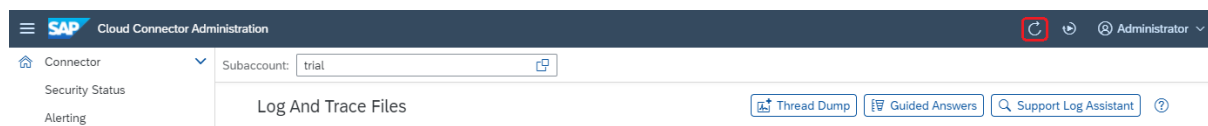
- [Guided Answers](#): A new tab or window opens, showing the Cloud Connector section in [Guided Answers](#). It helps you identify many issues that are classified through hierarchical topics. Once you found a matching issue, a solution is provided either directly, or by references to SAP Help Portal, Knowledge Base Articles (KBAs), and SAP notes.
- [Support Log Assistant](#): Opens the support log assistant. There, you can upload Cloud Connector log files and have them analyzed. After triggering the scan, the tool lists all issues for which a solution can be identified.

Note

The support log assistant analyzes the complete log. Therefore, also older issues may be found that are no longer relevant.

Once a problem has been identified, you should turn off the trace again by editing the trace and log settings accordingly to not flood the files with unnecessary entries.

Use the [Refresh](#) button to update the information that appears. For example, you can use this button because more trace files might have been written since you last updated the display.



Error Analysis and Support: Which Logs are Relevant?

If you contact SAP support for help, please always attach the appropriate log files and provide the timestamp or period, when the reported issue was observed. Depending on the situation, different logs may help to find the root cause.

Some typical settings to get the required data are listed below:

- **<Cloud Connector Loggers>** provide **details related to connections to SAP BTP and to backend systems as well as master-shadow communication in case of a high availability setup**. However, it does not contain any payload data. This kind of trace is written into `ljs_trace.log`, which is the most relevant log for the Cloud Connector.
- **<Other Loggers>** provide **details related to the tomcat runtime**, in which the Cloud Connector is running. The traces are written into `ljs_trace.log` as well, but they are needed only in very special support situations. If you don't need these traces, leave the level on `Information` or even lower.
- **Payload data** are written into the traffic trace file for HTTP or RFC requests if the payload trace is activated, or into the CPI-C trace file for RFC requests, if the CPI-C trace is set to level 3.
- **Payload SNC data** are written into a payload SNC trace file for incoming SNC RFC requests if SNC payload trace is activated.
- **<TLS trace>** is helpful to **analyze TLS handshake failures** from Cloud Connector to Cloud or from Cloud Connector to backend. It should be turned off again as soon as the issue has been reproduced and recorded in the traces.
- Setting the audit log on level `ALL` for **<Subaccount Audit Level>** is the easiest way to **check if a request reached the the Cloud Connector and if it is being processed**.

Related Information

[Getting Support](#)

1.2.3.12 Process Guidelines for Hybrid Scenarios

A hybrid scenario is one, in which applications running on SAP BTP require access to on-premise systems. Define and document your scenario to get an overview of the required process steps.

Tasks

[Document the Landscape of a Hybrid Solution \[page 700\]](#)

[Document Administrator Roles \[page 700\]](#)

[Document Communication Channels \[page 701\]](#)

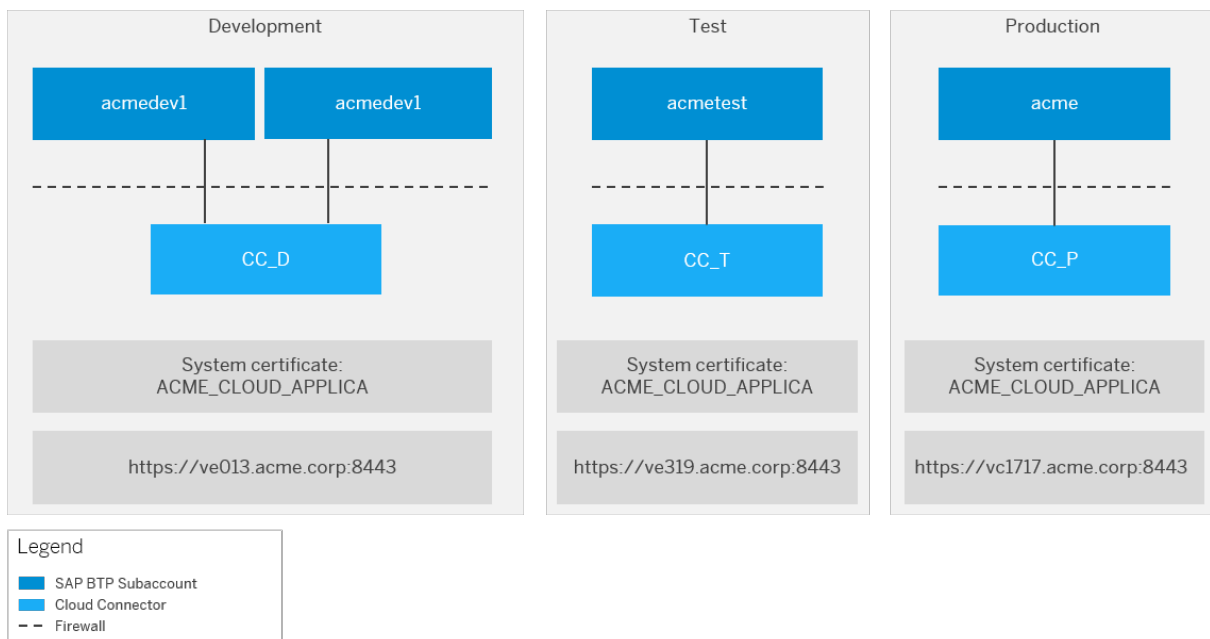
[Define Project and Development Guidelines \[page 701\]](#)

[Define How to Set a Cloud Application Live \[page 702\]](#)

Document the Landscape of a Hybrid Solution

To gain an overview of the cloud and on-premise landscape that is relevant for your hybrid scenario, we recommend that you diagrammatically document your cloud subaccounts, their connected Cloud Connectors and any on-premise back-end systems. Include the subaccount names, the purpose of the subaccounts (dev, test, prod), information about the Cloud Connector machines (host, domains), the URLs of the Cloud Connectors in the landscape overview document, and any other details you might find useful to include.

An example of landscape overview documentation could look like this:



Back to [Tasks \[page 699\]](#)

Document Administrator Roles

Document the users who have administrator access to the cloud subaccounts, to the Cloud Connector operating system, and to the Cloud Connector administration UI.

Such an administrator role documentation could look like following sample table:

Resource	bernardo@acme.com	mary@acme.com	smitha@acme.com	greg@acme.com
Cloud Subaccount (CA) Dev1	X			
CA Dev2		X		

Resource	bernardo@acme.com	mary@acme.com	smitha@acme.com	greg@acme.com
CA Test			X	X
CA Prod				X
Cloud Connector Dev1 + Dev2	X	X		
Cloud Connector Test			X	X
Cloud Connector Prod				X
Cloud Connector Dev1 + Dev2 file system			X	X
Cloud Connector Test file system				X
Cloud Connector Prod file system				

Back to [Tasks \[page 699\]](#)

Document Communication Channels

Create and document separate email distribution lists for both the cloud subaccount administrators and the Cloud Connector administrators.

An example of the documented communication channels could look like this:

Landscape	Distribution List
Cloud Subaccount Administrators	DL ACME BTP Subaccount Admins
Cloud Connector Administrators	DL ACME Cloud Connector Admins

Back to [Tasks \[page 699\]](#)

Define Project and Development Guidelines

Define and document mandatory project and development guidelines for your SAP BTP projects. An example of such a guideline could be similar to the following.

Every SAP BTP project in this organization requires the following:

- Use Maven, Nexus, Git-&-Gerrit for the application development.
- Align with accountable manager in projects (including the names).
- Align with accountable security officer in projects (including the names).

- For externally developed source code, an official handover to the organization.
- Fulfill connection restrictions in a three-system landscape, that is, use a staged landscape for dev, test and prod, and, for example, the dev landscape connects only to dev systems, and so on.
- Productive subaccounts cannot use the same Cloud Connector as a dev or test subaccount.

Back to [Tasks \[page 699\]](#)

Define How to Set a Cloud Application Live

Define and document how to set a cloud application live and how to configure needed connectivity for such an application.

For example, the following processes could be seen as relevant and should be defined and document in more detail:

1. Transferring application to production: Steps for transferring an application to the productive status on the SAP BTP.
2. Application connectivity: The steps for adding a connectivity destination to a deployed application for connections to other resources in the test or productive landscape.
3. Cloud Connector Connectivity: Steps for adding an on-premise resource to the Cloud Connector in the test or productive landscapes to make it available for the connected cloud subaccounts.
4. On-premise system connectivity: The steps for setting up a trusted relationship between an on-premise system and the Cloud Connector, and to configure user authentication and authorization in the on-premise system in the test or productive landscapes.
5. Application authorization: The steps for requesting and assigning an authorization that is available inside the SAP BTP application to a user in the test or productive landscapes.
6. Administrator permissions: Steps for requesting and assigning the administrator permissions in a cloud subaccount to a user in the test or productive landscape.

Back to [Tasks \[page 699\]](#)

1.2.3.13 Configuring Backup

Find an overview of backup procedures for the Cloud Connector.

Configuration Backup [page 623]	Backup and restore your Cloud Connector configuration via the administration UI.
Backup [page 531]	Manage the Cloud Connector's configuration backup via REST API.
Backup And Restore Configuration [page 605]	Example: Backup and restore the Cloud Connector configuration via REST API.

1.2.3.14 Secure Store

Reduce the size of the Cloud Connector Secure Store.

The Secure Store is a container for confidential information that is a part of the Cloud Connector configuration. The Secure Store resides in file `SSFS_SCC.DAT`, located in directory `scc_config`.

Every change affecting the Secure Store (such as changing the proxy password) is incremental, that is, it is appended to the Secure Store. As a consequence, the Secure Store will grow over time.

At some point it is sensible to remove obsolete entries to reduce the size of the Secure Store, and hence the size of the file `SSFS_SCC.DAT`.

To do this, choose **Configuration > Shrink Secure Store** (top right corner of the screen) from the Cloud Connector main menu.

1.2.4 Security

Learn how Cloud Connector features help you manage security.

Features

Security is a crucial concern for any cloud-based solution. It has a major impact on the business decision of enterprises whether to make use of such solutions. SAP BTP is a platform-as-a-service offering designed to run business-critical applications and processes for enterprises, with security considered on all levels of the on-demand platform:

Level	Features
Application Layer [page 704]	<ul style="list-style-type: none">• Frontend security• Security standard-based application development
Service Layer [page 705]	<ul style="list-style-type: none">• Identity and access management• Data protection• Regulatory compliance management
Cloud Infrastructure Layer [page 710]	<ul style="list-style-type: none">• Network infrastructure and communication• Sandboxing• Intrusion detection and prevention
Physical and Environmental Layer [page 711]	<ul style="list-style-type: none">• Strict physical access control• High availability

The Cloud Connector enables integration of cloud applications with services and systems running in customer networks, and supports database connections from the customer network to SAP HANA databases running on SAP BTP. As these are security-sensitive topics, this section gives an overview on how the Cloud Connector helps maintain security standards for the mentioned scenarios.

Target Audience

The content of this section concerns:

- System and IT administrators
- Technology consultants
- Solution architects

Related Information

[Security Guidelines \[page 711\]](#)

1.2.4.1 Application Layer

On application level, the main tasks to ensure secure Cloud Connector operations are to provide appropriate frontend security (for example, validation of entries) and a secure application development.

Product Security Standard

Basically, you should follow the rules given in the product security standard, for example, protection against cross-site scripting (XSS) and cross-site request forgery (XSRF).

Scope and Design

The scope and design of security measures on application level strongly depend on the specific needs of your application.

1.2.4.2 Service Layer

You can use SAP BTP Connectivity to securely integrate cloud applications with systems running in isolated customer networks.

Overview

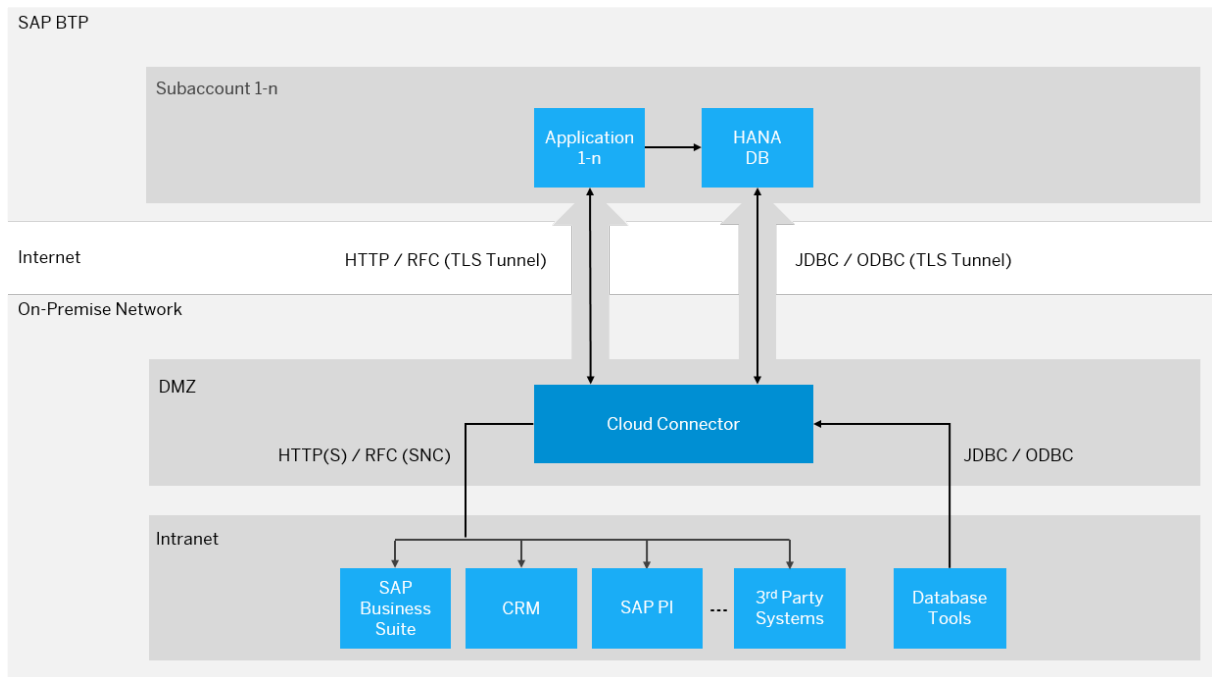
After installing the Cloud Connector as integration agent in your on-premise network, you can use it to establish a persistent TLS tunnel to SAP BTP subaccounts.

To establish this tunnel, the Cloud Connector administrator must authenticate himself or herself against the related SAP BTP subaccount of which he or she must be a member. Once established, the tunnel can be used by applications of the connected subaccount to remotely call systems in your network.

Architecture

The figure below shows a system landscape in which the Cloud Connector is used for secure connectivity between SAP BTP applications and on-premise systems.

- A single Cloud Connector instance can connect to multiple SAP BTP subaccounts, each connection requiring separate authentication and defining an own set of configuration.
- You can connect an arbitrary number of SAP and non-SAP systems to a single Cloud Connector instance.
- The on-premise system does not need to be touched when used with the Cloud Connector, unless you configure trust between the Cloud Connector and your on-premise system. A trust configuration is required, for example, for principal propagation (single sign-on), see [Configuring Principal Propagation \[page 420\]](#).
- You can operate the Cloud Connector in a high availability mode. To achieve this, you must install a second (redundant) Cloud Connector (shadow instance), which takes over from the master instance in case of a downtime.
- The Cloud Connector also supports the communication direction from the on-premise network to the SAP BTP subaccount, using a database tunnel that lets you connect common ODBC/JDBC database tools to SAP HANA as well as other available databases in SAP BTP.



Related Information

[Network Zones \[page 706\]](#)

[Inbound Connectivity \[page 707\]](#)

[Outbound Connectivity \[page 709\]](#)

[Audit Log \[page 709\]](#)

1.2.4.2.1 Network Zones

Choosing a network zone for the Cloud Connector installation.

A company network is usually divided into multiple network zones according to the security level of the contained systems. The DMZ network zone contains and exposes the external-facing services of an organization to an untrusted network, typically the Internet. Besides this, there can be one or multiple other network zones which contain the components and services provided in the company's intranet.

You can set up the Cloud Connector either in the DMZ or in an inner network zone. Technical prerequisites for the Cloud Connector to work properly are:

- The Cloud Connector must have access to the SAP BTP landscape host, either directly or via HTTPS proxy (see also: [Prerequisites \[page 351\]](#)).
- The Cloud Connector must have direct access to the internal systems it shall provide access to. I.e. there must be transparent connectivity between the Cloud Connector and the internal system.

It's a company's decision, whether the Cloud Connector is set up in the DMZ and operated centrally by an IT department or set up in the intranet and operated by the line of business.

Related Information

[Network Zones \[page 368\]](#)

1.2.4.2.2 Inbound Connectivity

For inbound connections into the on-premise network, the Cloud Connector acts as a reverse invoke proxy between SAP BTP and the internal systems.

Exposing Resources

Once installed, none of the internal systems are accessible by default through the Cloud Connector: you must configure explicitly each system and each service and resource on every system to be exposed to SAP BTP in the Cloud Connector.

You can also specify a virtual host name and port for a configured on-premise system, which is then used in the cloud. Doing this, you can avoid that information on physical hosts is exposed to the cloud.

TLS Tunnel

The TLS (Transport Layer Security) tunnel is established from the Cloud Connector to SAP BTP via a so-called **reverse invoke** approach. This lets an administrator have full control of the tunnel, since it can't be established from the cloud or from somewhere else outside the company network. The Cloud Connector administrator is the one who decides when the tunnel is established or closed.

The tunnel itself is using TLS with strong encryption of the communication, and mutual authentication of both communication sides, the client side (Cloud Connector) and the server side (SAP BTP).

The X.509 certificates which are used to authenticate the Cloud Connector and the SAP BTP subaccount are issued and controlled by SAP BTP. They are kept in secure storages in the Cloud Connector and in the cloud. Having encrypted and authenticated the tunnel, confidentiality and authenticity of the communication between the SAP BTP applications and the Cloud Connector is guaranteed.

Restricting Allowed Applications

As an additional level of control, the Cloud Connector optionally allows restricting the list of SAP BTP applications which are able to use the tunnel. This is useful in situations where multiple applications are deployed in a single SAP BTP subaccount while only particular applications require connectivity to on-premise systems.

Isolation on Subaccount Level

SAP BTP guarantees strict isolation on subaccount level provided by its infrastructure and platform layer. An application of one subaccount is not able to access and use resources of another subaccount.

Supported Protocols

The Cloud Connector supports inbound connectivity for HTTP and RFC, any other protocol is not supported.

- The payload sent via these protocols is encrypted on TLS/tunnel-level.
- For the route from the Cloud Connector to the on-premise systems, Cloud Connector administrators have the choice for each configured on-premise system whether to use HTTP, HTTPS, RFC or RFC over SNC.
- For HTTPS, you can configure a so-called system certificate in the Cloud Connector which is used for the trust relationship between the Cloud Connector and the connected on-premise systems.
- For RFC over SNC, you can configure an SNC PSE in the Cloud Connector respectively.

Principal Propagation

The Cloud Connector also supports principal propagation of the cloud user identity to connected on-premise systems (single sign-on). For this, the system certificate (in case of HTTPS) or the SNC PSE (in case of RFC) is mandatory to be configured and trust with the respective on-premise system must be established. Trust configuration, in particular for principal propagation, is the only reason to configure and touch an on-premise system when using it with the Cloud Connector.

Related Information

[Configuring Principal Propagation \[page 420\]](#)

1.2.4.2.3 Outbound Connectivity

The Cloud Connector supports the communication direction from the on-premise network to SAP BTP, using a database tunnel.

The database tunnel is used to connect local database tools via JDBC or ODBC to the SAP HANA DB or other databases on SAP BTP, for example, SAP Business Objects tools like Lumira, BOE or Data Services.

- The database tunnel only allows JDBC and ODBC connections from the Cloud Connector into the cloud. A reuse for other protocols is not possible.
- The tunnel uses the same security mechanisms as for the inbound connectivity:
 - TLS-encryption and mutual authentication
 - Audit logging

To use the database tunnel, two different SAP BTP users are required:

- A platform user (member of the SAP BTP subaccount) establishes the database tunnel to the HANA DB.
- A HANA DB user is needed for the ODBC/JDBC connection to the database itself. For the HANA DB user, the role and privilege management of HANA can be used to control which actions he or she can perform on the database.

Related Information

[Using Service Channels \[page 608\]](#)

1.2.4.2.4 Audit Log

As audit logging is a critical element of an organization's risk management strategy, the Cloud Connector provides audit logging for the complete record of access between cloud and Cloud Connector as well as of configuration changes done in the Cloud Connector.

Integrity Check

The written audit log files are digitally signed by the Cloud Connector so that they can be checked for integrity (see also: [Manage Audit Logs \[page 690\]](#)).

Alerting

The audit log data of the Cloud Connector can be used to alert Cloud Connector administrators regarding unusual or suspicious network and system behavior.

Additional Use Cases

- The audit log data can provide auditors with information required to validate security policy enforcement and proper segregation of duties.
- IT staff can use the audit log data for root-cause analysis following a security incident.

Related Information

[Audit Logging \[page 690\]](#)

1.2.4.3 Cloud Infrastructure Layer

Infrastructure and network facilities of the SAP BTP ensure security on network layer by limiting access to authorized persons and specific business purposes.

Isolated Network

The SAP BTP landscape runs in an isolated network, which is protected from the outside by firewalls, DMZ, and communication proxies for all inbound and outbound communications to and from the network.

Sandboxed Environments

The SAP BTP infrastructure layer also ensures that platform services, like the SAP BTP Connectivity, and applications are running isolated, in sandboxed environments. An interaction between them is only possible over a secure remote communication channel.

1.2.4.4 Physical and Environmental Layer

Learn about data center security provided for SAP BTP Connectivity.

SAP BTP runs in SAP-hosted data centers which are compliant with regulatory requirements. The security measures include, for example:

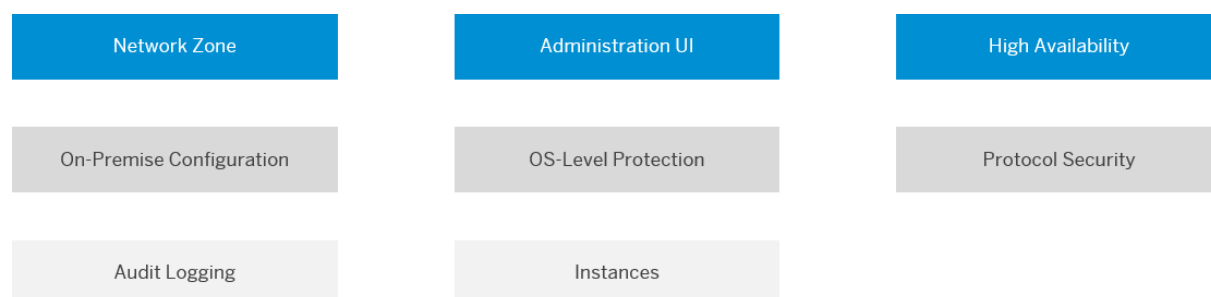
- strict physical access control mechanisms using biometrics, video surveillance, and sensors
- high availability and disaster recoverability with redundant power supply and own power generation

1.2.4.5 Security Guidelines

Find a checklist of recommended security measures for the Cloud Connector.

Topics

Hover over the elements for a description. Click an element to find the recommended actions in the table below.



- [#unique_186/unique_186_Connect_42_network](#) [page 712]
- [#unique_186/unique_186_Connect_42_ui](#) [page 712]
- [#unique_186/unique_186_Connect_42_availability](#) [page 713]
- [#unique_186/unique_186_Connect_42_protocols](#) [page 714]
- [#unique_186/unique_186_Connect_42_os](#) [page 712]
- [#unique_186/unique_186_Connect_42_oP](#) [page 714]
- [#unique_186/unique_186_Connect_42_instances](#) [page 714]
- [#unique_186/unique_186_Connect_42_audit](#) [page 713]

Recommended Actions

Topic	Description	Recommended Action
Network Zone Back to Topics [page 711]	Depending on the needs of the project, the Cloud Connector can be either set up in the DMZ and operated centrally by the IT department or set up in the intranet and operated by the line-of-business.	To access highly secure on-premise systems, operate the Cloud Connector centrally by the IT department and install it in the DMZ of the company network. Set up trust between the on-premise system and the Cloud Connector, and only accept requests from trusted Cloud Connectors in the system.
OS-Level Protection Back to Topics [page 711]	The Cloud Connector is a security-critical component that handles the inbound access from SAP BTP applications to systems of an on-premise network. Methods to secure the operating system , on which the Cloud Connector is running, should be applied.	Restrict access to the operating system on which the Cloud Connector is installed to the minimal set of users who should administrate the Cloud Connector. Use the machine which runs the Cloud Connector only for this purpose and don't reuse it for other scenarios. Use hard-drive encryption for the machine that runs the Cloud Connector. This ensures that the Cloud Connector configuration data cannot be read or modified by unauthorized users, even if they obtain access to the hard drive. Turn on the audit log on operating system level to monitor the file operations.
Administration UI Back to Topics [page 711]	After installation, the Cloud Connector provides an initial user name and password and forces the user (Administrator) to change the password upon initial logon.	Change the password of the Administrator user immediately after installation. Choose a strong password for the user (see also Recommendations for Secure Setup [page 383]). Configure a corporate LDAP system for the user management of the Cloud Connector administrator users. This guarantees that users of the Cloud Connector administration UI are named users and can be traced via the Cloud Connector audit log (see Use LDAP for User Administration [page 634]).

Topic	Description	Recommended Action
	<p>You can access the Cloud Connector administration UI remotely via HTTPS.</p> <p>After installation, it uses a self-signed X.509 certificate as TLS server certificate, which is not trusted by default by Web browsers.</p>	<p>Exchange the self-signed X.509 certificate of the Cloud Connector administration UI by a certificate that is trusted by your company and the company's approved Web browser settings (see Exchange UI Certificates in the Administration UI [page 629]).</p> <hr/> <p>For high-security scenarios, limit the access to the Cloud Connector administration UI to localhost (see also Recommendations for Secure Setup [page 383]).</p> <hr/> <p>Use a JVM that allows to limit the ciphers to a set considered safe (see Recommendations for Secure Setup [page 383]). A list of cipher suites, which are currently considered safe, is available in Use Secure Cipher Suites .</p>
<p>Audit Logging</p> <p>Back to Topics [page 711]</p>	<p>For end-to-end traceability of configuration changes in the Cloud Connector, as well as communication delivered by the Cloud Connector, switch on audit logging for productive scenarios.</p>	<p>Switch on audit logging in the Cloud Connector: set audit level to "All" (see Recommendations for Secure Setup [page 383] and Manage Audit Logs [page 690])</p> <hr/> <p>Cloud Connector administrators must ensure that the audit log files are properly archived and are not lost, to conform to the local regulations.</p> <hr/> <p>To gain end-to-end traceability, you should switch on audit logging also in the connected on-premise systems.</p>
<p>High Availability</p> <p>Back to Topics [page 711]</p>	<p>To guarantee high availability of the connectivity for cloud integration scenarios, run productive instances of the Cloud Connector in high availability mode, that is, with a second (redundant) Cloud Connector in place.</p>	<p>Use the high availability feature of the Cloud Connector for productive scenarios (see Install a Failover Instance for High Availability [page 652]).</p>

Topic	Description	Recommended Action
Supported Protocols Back to Topics [page 711]	<p>HTTP, HTTPS, RFC and RFC over SNC are currently supported as protocols for the communication direction from the cloud to on-premise.</p> <p>The route from the application VM in the cloud to the Cloud Connector is always encrypted.</p> <p>You can configure the route from the Cloud Connector to the on-premise system to be encrypted or unencrypted.</p>	<p>The route from the Cloud Connector to the on-premise system should be encrypted using TLS (for HTTPS) or SNC (for RFC).</p> <p>Trust between the Cloud Connector and the connected on-premise systems should be established (see Set Up Trust [page 420]).</p>
Configuration of On-Premise Systems Back to Topics [page 711]	<p>When configuring the access to an internal system in the Cloud Connector, map physical host names to virtual host names to prevent exposure of information on physical systems to the cloud.</p> <hr/> <p>When configuring the access to an internal system, restrict access to those resources which are actually required by the cloud applications. Do not expose the complete system.</p> <hr/> <p>To allow access only for trusted applications of your SAP BTP subaccount to on-premise systems, configure the list of trusted applications in the Cloud Connector.</p>	<p>Use hostname mapping of exposed on-premise systems in the access control of the Cloud Connector (see Configure Access Control (HTTP) [page 457] and Configure Access Control (RFC) [page 465]).</p> <hr/> <p>Narrow access to on-premise systems to resources required by the relevant cloud applications in the access control of the Cloud Connector (see Configure Access Control (HTTP) [page 457] and Configure Access Control (RFC) [page 465]).</p> <hr/> <p>Narrow the list of cloud applications which are allowed to use the on-premise tunnel to the ones that need on-premise connectivity (see Set Up Trust [page 420]).</p>
Cloud Connector Instances Back to Topics [page 711]	<p>You can connect a single Cloud Connector instance to multiple SAP BTP subaccounts.</p> <p>Subaccounts can be created by SAP BTP users as a self service. Different subaccounts are often used to separate development, test and production.</p> <p>Do not mix productive Cloud Connector usages with development or test scenarios.</p>	<p>Use different Cloud Connector instances to separate productive and non-productive scenarios.</p>

Related Information

[Recommendations for Secure Setup \[page 383\]](#)

[Secure the Activation of Traffic Traces \[page 659\]](#)

1.2.5 Upgrade

Upgrade your Cloud Connector and avoid connectivity downtime during the update.

The steps for upgrading your Cloud Connector are specific to the operating system that you use. Previous settings and configurations are automatically preserved.

⚠ Caution

Upgrade is supported only for *installer* versions, not for *portable* versions, see [Installation \[page 349\]](#). Before upgrading, please check the [Prerequisites \[page 351\]](#) and make sure your environment fits the new version. We recommend that you create a [Configuration Backup \[page 623\]](#) before starting an upgrade.

Avoid Connectivity Downtime

If you have a single-machine Cloud Connector installation, a short downtime is unavoidable during the upgrade process. However, if you have set up a master and a shadow instance, you can perform the upgrade without downtime by executing the following procedure:

1. Shut down the shadow instance.
2. Perform the upgrade on the shadow instance. (Follow the relevant procedure below.)
3. Restart the shadow instance and wait until it has connected to the master instance.
4. Perform a [Switch Roles](#) operation by pressing the corresponding button in the master administration UI. The master instance has now changed into a shadow instance.

⚠ Caution

After upgrading the former shadow instance from a version prior to 2.13 and having switched its role to be the new master instance, reset high availability settings in *both* instances *now*, before continuing to upgrade the second instance from a version prior to 2.13 as well. The master-shadow connection must be re-established after both instances have been upgraded from versions prior to 2.13 to versions 2.13 or higher.

5. Shut down the new shadow instance and perform the upgrade procedure on it as well.
6. Restart the new shadow instance and wait until it has connected to the already upgraded current master instance.
7. Perform again the [Switch Roles](#) operation if you want the previous master instance to act as the new master instance again.

Result: Both instances have now been upgraded without connectivity downtime and without configuration loss.

For more information, see [Install a Failover Instance for High Availability \[page 652\]](#).

Microsoft Windows OS

1. Uninstall the Cloud Connector as described in [Uninstallation \[page 718\]](#) and make sure to retain the existing configuration.
2. Reinstall the Cloud Connector within the same directory. For more information, see [Installation on Microsoft Windows OS \[page 375\]](#).
3. Before accessing the administration UI, clear your browser cache to avoid any unpredictable behavior due to the upgraded UI.

Linux OS

1. Execute the following command: :

```
rpm -U com.sap.scc-ui-<version>.rpm
```

Note

Daemon extensions (as of Cloud Connector version 2.12.3)

All extensions to the daemon provided via `scc_daemon_extension.sh` mechanism will survive a version update. An upgrade to version 2.12.3 will already consider an existing file, even though previous versions were not supporting that feature.

2. Before accessing the administration UI, clear your browser cache to avoid any unpredictable behavior due to the upgraded UI.

1.2.6 Update the Java VM

How to update the Java VM used by the Cloud Connector.

Sometimes you must update the Java VM used by the Cloud Connector, for example, because of expired TLS certificates contained in the JVM trust store, bug fixes, deprecated JVM versions, and so on.

- If you make a replacement in the same directory, shut down the Cloud Connector, upgrade the JVM, and restart the Cloud Connector when you are done.
- **If you change the installation directory** of the JVM, **follow the steps below** for your operating system.

Make sure that the JVM has been installed successfully.

Note

A Java Runtime Environment (JRE) is not sufficient. You must use a JDK or SAP JVM.

Microsoft Windows OS

1. Make sure that the current user has administrative privileges.
2. Shutdown the Cloud Connector (for example, by stopping the corresponding Windows service or by double-clicking the *Stop SAP Cloud Connector* shortcut).
3. Open the *registry editor* (regedit32) and locate the following registry entry:
HKEY_LOCAL_MACHINE\SOFTWARE\SAP\Cloud Connector.
4. Change the value `JavaHome` to reflect the installation directory of the new SAP JVM or JDK.

Note

The `bin` subdirectory must not be part of the `JavaHome` value.

If the `JavaHome` value does not yet exist, create it here with a "String Value" (REG_SZ) and specify the full path of the Java installation directory, for example: `C:\Program Files\sapjvm`.

5. Close the registry editor and restart the Cloud Connector.

Linux OS

1. Open a shell command line prompt as root user.
2. In this shell, set the environment variable `JAVA_HOME`, pointing to the installation directory of the new SAP JVM or JDK, for example:
 - in `csh/tcsh`:

```
setenv JAVA_HOME /opt/sap/sapjvm_8
```

- in `sh/bash/dash/zsh`:

```
export JAVA_HOME=/opt/sap/sapjvm_8
```

3. Execute the command

```
System V init distributions: service scc_daemon stop  
systemd distributions: systemctl stop scc_daemon
```

4. Execute the command

```
System V init distributions: /opt/sap/scc/daemon.sh reinstall  
systemd distributions: /opt/sap/scc/daemon.sh reinstallSystemd
```

5. Execute the command

```
System V init distributions: service scc_daemon start
```

```
systemd distributions: systemctl start scc_daemon
```

Both Operating Systems

Note

- If you use your own CA certificates for the Email configuration (see [Alerting \[page 687\]](#)) or for LDAP (see [Use LDAP for User Administration \[page 634\]](#)), you must reimport them to the JVM trust store as described there.
- Make sure the selected cipher suites are accepted by the JVM you are about to install. If in doubt, revert to the default selection prior to changing the JVM.

After executing the above steps, the Cloud Connector should be running again and should have picked up the new Java version during startup. You can verify this by logging in to the Cloud Connector with your favorite browser, opening the *About* dialogue and checking that the field `<Java Details>` shows the version number and build date of the new Java VM. After you verified that the new JVM is indeed used by the Cloud Connector, delete or uninstall the old JVM.

1.2.7 Uninstallation

Uninstall an installer version or portable version of the Cloud Connector.

- If you have installed an installer variant of the Cloud Connector, follow the steps for your operating system to uninstall the Cloud Connector.
- To uninstall a developer version, proceed as described in section **Portable Variants**.

Microsoft Windows OS

1. In the Windows software administration tool, search for `Cloud Connector` (formerly named `SAP HANA cloud connector 2.x`).
2. Select the entry and follow the appropriate steps to uninstall it.
3. When you are uninstalling in the context of an upgrade, make sure to retain the configuration files.

Linux OS

To uninstall Cloud Connector 2.x, execute the following command:

```
rpm -e com.sap.scc-ui
```

⚠ Caution

This command also removes the configuration files.

Mac OS X

There is no installer variant for Mac OS X, only a `portable` one.

Portable Variants

(Microsoft Windows OS, Linux OS, Mac OS X) If you have installed a portable version (zip or tgz archive) of the Cloud Connector, simply remove the directory in which you have extracted the Cloud Connector archive.

Related Information

[Installation \[page 349\]](#)

1.2.8 Frequently Asked Questions

Answers to the most common questions about the Cloud Connector.

Technical Issues

Does the Cloud Connector send data from on-premise systems to SAP BTP or the other way around?

The connection is opened from the on-premise system to the cloud, but is then used in the other direction.

An on-premise system is, in contrast to a cloud system, normally located behind a restrictive firewall and its services aren't accessible thru the Internet. This concept follows a widely used pattern often referred to as *reverse invoke proxy*.

Is the connection between the SAP BTP and the Cloud Connector encrypted?

Yes, by default, TLS encryption is used for the tunnel between SAP BTP and the Cloud Connector.

If used properly, TLS is a highly secure protocol. It is the industry standard for encrypted communication and also, for example, as a secure channel in HTTPS.

Keep your Cloud Connector installation and JDK updated to avoid the use of weak and deprecated ciphers for TLS communication. Which cipher and TLS versions are actually used, is defined by both the cloud region setup and the JDK that is used for Cloud Connector. The TLS implementation used for the communication is the one of the JDK.

Can I use a TLS-terminating firewall between Cloud Connector and SAP BTP?

This is not possible. Basically, this is a desired man-in-the-middle attack, which does not allow the Cloud Connector to establish a mutual trust to the SAP BTP side.

Can I copy/clone a Cloud Connector installation and use it in parallel on a different machine?

This is not supported. You would face issues regularly, as those two instances will be considered as one, which is not expected by the cloud side.

If you just want to move the installation to a new machine, create a backup via the [Configuration Backup \[page 623\]](#) feature, create a new installation, import the backup, and discard the previous installation.

What is the oldest version of SAP Business Suite that's compatible with the Cloud Connector?

The Cloud Connector can connect an SAP Business Suite system version 4.6C and newer.

Which Java versions are supported to run the Cloud Connector?

		Supported Java Version				
		6	7	8	11	17
Cloud Connector Version	< 2.7.2	Yes	Yes	No	No	No
	= 2.7.2	Yes	Yes	Yes	No	No
	>= 2.8	No	Yes	Yes	No	No
	>= 2.12.3	No	No	Yes	No	No
	>= 2.14.0	No	No	Yes	Yes	No
	>= 2.15.0	No	No	Yes	Yes	Yes

⚠ Restriction

Support for Java 7 has been discontinued. For more information, see [Prerequisites \[page 352\]](#).

→ Tip

We recommend that you always use the latest patch level of the respective Java version.

⚠ Caution

Version 2.8 and later of the Cloud Connector may have problems with ciphers in Google Chrome, if you use the JVM 7. For more information read [this SCN Article](#).

Which configuration in the SAP BTP destinations do I need to handle the user management access to the Cloud User Store of the Cloud Connector?

See [Configure an On-Premise User Store](#).

Is the Cloud Connector sufficient to connect the SAP BTP to an SAP ABAP back end or is SAP BTP Integration needed?

It depends on the scenario: For pure point-to-point connectivity to call on-premise functionality like BAPIs, RFCs, OData services, and so on, that are exposed via on-premise systems, the Cloud Connector might suffice.

However, if you require advanced functionality, for example, n-to-n connectivity as an integration hub, SAP BTP Integration – Process Integration is a more suitable solution. SAP BTP Integration can use the Cloud Connector as a communication channel.

How much bandwidth does the Cloud Connector consume?

The amount of bandwidth depends greatly on the application that is using the Cloud Connector tunnel. If the tunnel isn't currently used, but still connected, a few bytes per minute is used simply to keep the connection alive.

What happens to a response if there's a connection failure while a request is being processed?

The response is lost. The Cloud Connector only provides tunneling, it does not store and forward data when there are network issues.

Where should I install the Cloud Connector?

For productive instances, we recommend installing the Cloud Connector on a single purpose machine. This is relevant for [Security \[page 703\]](#). For more details on which network zones to choose for the Cloud Connector setup, see [Network Zones \[page 368\]](#).

How does a disaster recovery setup look like for the Cloud Connector?

There is no explicit implementation of a disaster recovery setup for the Cloud Connector. However, it is actually not needed.

Instead, make sure you have machines available in some other data center than the one in which your productive setup is running. Also, make sure you regularly generate a [Configuration Backup \[page 623\]](#).

If a disaster situation occurs, install the Cloud Connector again and restore the latest backup. Immediately after the restart that is required after restoring the backup, you are back to a running setup, as long as all the backend systems are reachable from the new location.

How many servers do I need to deploy the Cloud Connector?

We recommend that you use at least three servers, with the following purposes:

- Development
- Production master
- Production shadow

Note

Do not run the production master and the production shadow as VMs inside the same physical machine. Doing so removes the redundancy, which is needed to guarantee high availability. A QA (Quality Assurance) instance is a useful extension. For disaster recovery, you will need two additional instances: another master instance, and another shadow instance as a reserve for the disaster case.

What are the hardware requirements to deploy the Cloud Connector?

See: [Prerequisites \[page 351\]](#).

Is NTLM supported for authorization against the proxy server?

No, the Cloud Connector currently supports only basic authentication.

Which operating systems are supported by the Cloud Connector?

See [Prerequisites \[page 351\]](#).

Which processor architectures are supported by the Cloud Connector?

We currently support 64-bit operating systems running only on an x86-64 processor (also known as x64, x86_64 or AMD64), and for Linux also on the *PowerPC Little Endian* variant (also known as *ppc64le*).

See: [Prerequisites \[page 351\]](#).

Can I use the Cloud Connector without an ABAP back end?

Yes, you should be able to connect almost any system that supports the HTTP Protocol, to the SAP BTP, for example, Apache HTTP Server, Apache Tomcat, Microsoft IIS, or Nginx.

Can I authenticate with client certificates configured in SAP BTP destinations at HTTP services that are exposed via the Cloud Connector?

No, this is not possible. For client certificate authentication, an end-2-end TLS communication is required. This is not the case, because the Cloud Connector needs to inspect incoming requests in order to perform access control checks.

How can I do connection pooling for HTTP services that are exposed via the Cloud Connector?

The Cloud Connector itself does not perform connection pooling, but provides a 1-to-1 mapping for each logical connection received through the tunnel.

By this mapping, a new connection to the backend system is opened, and kept open until closed either by the backend or by the client on cloud side.

The actual connection pooling is defined by the application client on cloud side:

- If a connection is re-used in the client library, it is re-used on the Cloud Connector side as well.
- If it is closed immediately, also the mapped one on Cloud Connector side will be closed immediately.

Can I open two windows or tabs in a single browser instance to administrate the Cloud Connector?

No, this is not supported and may cause odd behavior on the different screens, in particular when trying to navigate through multiple subaccounts. If you like to open the administration UI twice, use two separate browser instances.

Does the Cloud Connector delete or modify HTTP headers?

Modifications of HTTP response headers are done if needed. In particular, Set-Cookie domains are adjusted according to the configured domain and host mappings. Also, in case of redirects, the location header will be adjusted according to the host mappings. Modifications of HTTP request headers are also done if needed, which is currently only the case for the *Host* header content. It will be replaced by the internal host, if the host mapping configuration is set up accordingly. The Cloud Connector will not delete any header that is sent by the cloud application. However, the Connectivity service will drop Connectivity service-specific headers, such as `SAP-Connectivity-Authentication` or `SAP-Connectivity-ConsumerAccount` so that those headers will neither reach the Cloud Connector nor the eventual backend.

Administration

Are there Audit Logs for changes in the Cloud Connector?

Yes, find more details here: [Manage Audit Logs \[page 690\]](#).

Is it possible to split authorization?

No, currently there is only one role that allows complete administration of the Cloud Connector.

Can I configure multiple administrative subaccounts?

Yes, to enable this, you must configure an LDAP server. See: [Use LDAP for User Administration \[page 634\]](#).

How can I reset the Cloud Connector's administrator password when not using LDAP for authentication?

Visit <https://tools.hana.ondemand.com/#cloud> to download the portable version of the Cloud Connector. Extract the *users.xml* file in the *config* directory to the *config* directory of your Cloud Connector installation, then restart the Cloud Connector.

This resets the password and user name to their default values.

You can manually edit the file; however, we strongly recommend that you use the *users.xml* file.

How do I create a backup of the Cloud Connector configuration?

Starting with Cloud Connector version 2.11, you can use a dedicated backup feature, either from the administration UI (see [Configuration Backup \[page 623\]](#)) or via REST API (see [Backup \[page 531\]](#)).

Can I create a backup of the complete installation?

Yes, you can create an archive file of the installation directory to create a full backup. Before you restore from a backup, note the following:

- If you restore the backup on a different host, the UI certificate will be invalidated.
- Before you restore the backup, you should perform a "normal" installation and then replace the files. This registers the Cloud Connector at your operating systems package manager.

Why do I need a user ID during configuration?

This user opens the tunnel and generates the certificates that are used for mutual trust later on.

The user is not part of the certificate that identifies the Cloud Connector.

In both the Cloud Connector UI and in the SAP BTP cockpit, this user ID appears as the one who performed the initial configuration (even though the user may have left the company).

What happens to a Cloud Connector connection if the user who created the tunnel leaves the company?

This does not affect the tunnel, even if you restart the Cloud Connector.

What do changes in major or minor version numbers mean?

The semantics of Cloud Connector versions are explained in detail [here](#).

Does SAP provide support for older Cloud Connector versions?

SAP supports the latest 2 feature releases in parallel. For the latest feature release, the last 2 patch levels are supported. For the previous feature release, only its latest patch level is supported.

For more information on the Cloud Connector support strategy, see SAP Note [3302250](#).

What is the difference between “subaccount name” and “subaccount user”?

SAP BTP customers can purchase subaccounts and deploy applications into these subaccounts.

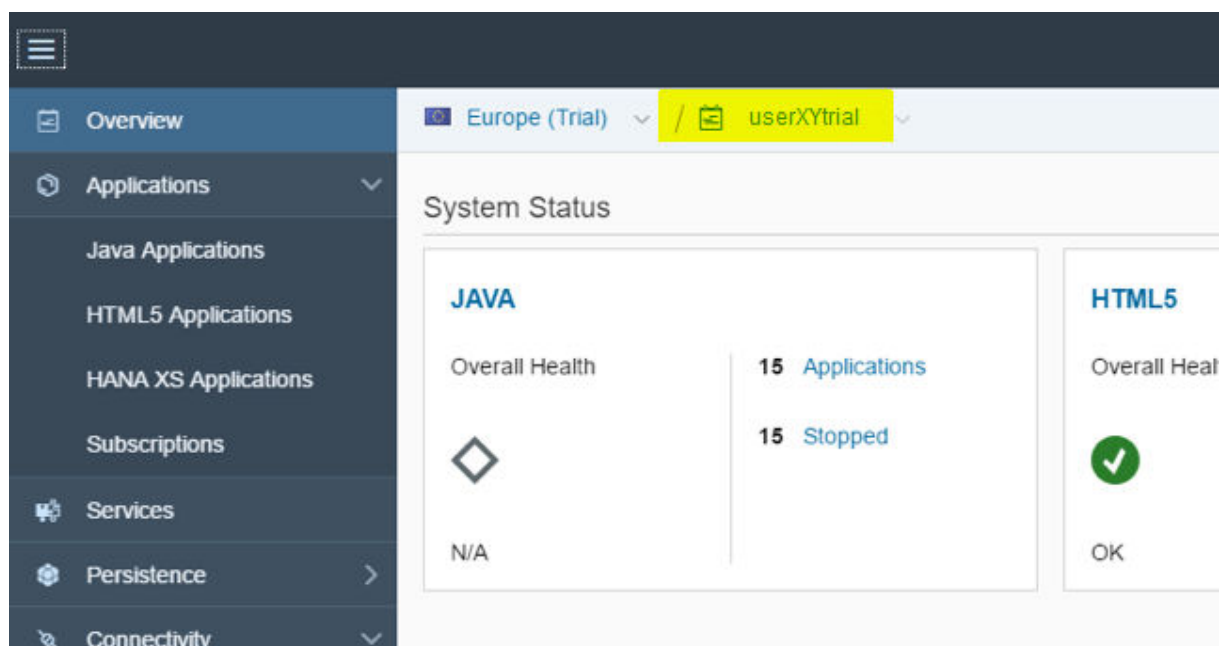
Additionally, there are users, who have a password and can log in to the cockpit and manage all subaccounts they have permission for.

- A single subaccount can be managed by multiple users, for example, your company may have several administrators.
- A single user can manage multiple subaccounts, for example, if you have multiple applications and want them (for isolation reasons) to be split over multiple subaccounts.

Find your account name by taking the following steps:

1. Open the SAP BTP cockpit.
2. Log in with your subaccount user.
3. You'll see the subaccount name in the top left section of the screen.

For trial users, the account name is typically your user name, followed by the suffix “trial”:



Features

Does the Cloud Connector work with the SAP BTP Cloud Foundry environment?

As of version 2.10, the Cloud Connector can establish a connection to regions based on the SAP BTP Cloud Foundry environment. Newer regions, however, require a Cloud Connector version 2.11 or higher.

Does the Cloud Connector work with SAP S/4HANA Cloud?

As of version 2.10, the Cloud Connector offers a Service Channel to S/4HANA Cloud instances, given that they are associated with the respective SAP BTP subaccount. For more information, see [Using Service Channels \[page 608\]](#).

Also supported as of version 2.10: S/4HANA Cloud communication scenarios invoking HTTP services or remote-enabled function modules (RFMs) in on-premise ABAP systems.

Does the Cloud Connector work with the SAP BTP ABAP environment?

As of version 2.11, the Cloud Connector supports communication from and to the SAP BTP ABAP environment, when using the **Neo** Connectivity service. Using the **Cloud Foundry** Connectivity service requires a Cloud Connector version 2.12.3 or higher.

How do I bind multiple Cloud Connectors to one SAP BTP subaccount?

As of version 2.9, you can connect multiple Cloud Connectors to a single subaccount. This lets you assign multiple separate corporate network segments.

Those Cloud Connectors are distinguishable based on the location ID, which you must provide to the destination configuration on the cloud side.

Note

During an upgrade, location IDs provided in earlier versions of the Cloud Connector are dropped to ensure that running scenarios are not disturbed.

Is WebSocket communication through the Cloud Connector supported?

Yes, this is possible as of version 2.12.

Is there any plan to add traffic management functionality in Cloud Connector?

No, this functionality is not currently planned.

Can I use the Cloud Connector from cloud to on-premise for any protocol?

As of version 2.10, you can use the TCP channel of the Cloud Connector, if the client supports a SOCKS5 proxy to establish the connection. However, only the HTTP and RFC protocols currently provide an additional level of access control by checking invoked resources.

Can I use the Cloud Connector from on-premise to cloud for any protocol?

This is possible only for a limited set of protocols. You can use the Cloud Connector as a JDBC or ODBC proxy to access the HANA DB instance within your SAP BTP Neo subaccount (service channel). This is sometimes referred to as “HANA protocol”. Also, there are service channels for SSH access to SAP BTP Neo virtual machines, and for RFC access to ABAP cloud systems. All of these service channels provide access to endpoints that are not visible in the Internet.

For HTTP, the endpoints that could be addressed are visible in the Internet. Therefore, you can simply use your normal network infrastructure that is prepared for accessing HTTPS endpoints in the Internet anyway.

Can I check the communication of the service channel?

No, the audit log monitors access only from SAP BTP to on-premise systems.

Troubleshooting

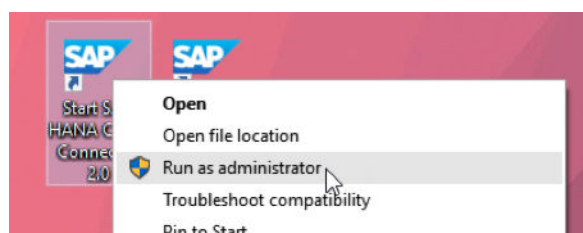
How do I fix the “Could not open Service Manager” error message?

You are probably seeing this error message due to missing administrator privileges. Right-click the cloud connector shortcut and select Run as administrator.

If you don't have administrator privileges on your machine you can use the portable variant of the Cloud Connector.

Note

The portable variants of the Cloud Connector are meant for nonproductive scenarios only.



How do I set JAVA_HOME and PATH correctly?

`JAVA_HOME` must point to the installation directory of your SAP JVM or JDK while `PATH` must contain the `bin` folder inside the installation directory of your SAP JVM or JDK. This is relevant in particular for the portable versions. The installers will also detect JDKs in other places.

How do I use the various .bat-Batch or .sh-Shell script tools in the Cloud Connector directory?

Open a command line prompt with administrator privileges or with sufficient user privileges to read and write files in the Cloud Connector directory. Then, make sure the environment variable `JAVA_HOME` is set to the installation directory of the JDK used by the Cloud Connector.

Afterwards, switch to the Cloud Connector directory and call the appropriate batch or shell script tools via `<toolname>.bat` or `./<toolname>.sh`. If the respective tool script requires further input parameters, its usage syntax will be written to the console.

When I try to open the Cloud Connector UI, Google Chrome opens a Save as dialog, Firefox displays some cryptic signs, and Internet Explorer shows a blank page, how do I fix this?

This happens when you try to access the Cloud Connector over HTTP instead of HTTPS. HTTP is the default protocol for most browsers.

Adding "https://" to the beginning of your URL should fix the problem. For localhost, you can use `https://localhost:8443/`.

1.2.9 REST APIs

Find general information on the Cloud Connector REST APIs.

The Cloud Connector provides REST APIs to support automation of configuration and monitoring tasks. REST APIs are exposed on the same host and port that you use to access to the Cloud Connector.

[Request and Response Payload Format \[page 728\]](#)

[Security and Session \[page 729\]](#)

[User Roles \[page 730\]](#)

[Return Codes \[page 730\]](#)

[Hypertext Application Language \[page 731\]](#)

[Available REST APIs \[page 731\]](#)

[Sample Code \[page 731\]](#)

Request and Response Payload Format

The payload (that is, the data transmitted in the body) of requests and responses is mostly coded in JSON format. The following example shows the request payload `{description:<value>}` coded in JSON:

```
{"description": "very helpful description"}
```

Values that represent a date are provided as a UTC long number, which is the number of milliseconds since 1 January 1970 00:00:00 UT (GMT+00).

In case of errors, the HTTP status code is `4xx` or `500`. Error details are supplied in the response body in JSON format:

```
{type: <type>, message: <message>}
```

where `type` can be one of the following strings:

```
INVALID_REQUEST, ILLEGAL_STATE, NOT_FOUND, INVALID_CONFIGURATION,  
RUNTIME_FAILURE, SHADOW_UPDATE, FORBIDDEN_REQUEST
```

The request JSON is listed in the API descriptions in an abbreviated form, showing only the property names and omitting the values. Details on the values (data types and restrictions) are provided separately below the respective API descriptions.

The API documentation also includes possible error types, and details on those errors below the API description. These errors pertain to property values of the payload only. We do not claim this list of errors to be exhaustive. In particular, errors caused by obviously erroneous input, such as missing mandatory or malformed property values, may have been omitted. As a rule of thumb, missing or invalid values result in `INVALID_REQUEST`. Errors caused elsewhere (for example, inappropriate header field values) are not listed.

Note

Request bodies in JSON format require the header field `Content-Type application/json`. The API descriptions do not explicitly mention this fact. Header fields are shown only if they deviate from the default `Content-Type: application/json`.

Back to [Top \[page 728\]](#)

Security and Session

The Cloud Connector supports the authentication types *basic authentication* and *form-based authentication*. Once authenticated, a client can keep the session and execute subsequent requests in the session. A session avoids the overhead caused by authentication. As usual, the session ID can be obtained from the response header field `Set-cookie` (as `JSESSIONID=<session Id>`), and must be sent in the request header `Cookie`: `JSESSIONID=<session Id>`.

The Cloud Connector employs CSRF tokens to counter CSRF (cross-site request forgery) attacks. Upon first request, a CSRF token is generated and sent back in the response header in field `X-CSRF-Token`. The client

application must retain this token and send it in all subsequent requests as header field *X-CSRF-Token* together with the session cookie as explained above.

Note

No CSRF token is generated if request header field *Connection* has the value *close* (as opposed to *keep-alive*). In other words, if you want to make stateful, session-based REST calls, use *Connection: keep-alive*, extract the CSRF token from the first response header and subsequently include it in all request headers. Otherwise, use *Connection: close* and always submit user and password through *basic authentication*.

An inactive session will incur a timeout at some point, and will consequently be removed. A request using an expired session will receive a login page (*Content-type: text/html*). The status code of the response is 200 in this case. Hence, the only way to detect an expired session is to pay attention to the content type and status code. Content type *text/html* in a connection with status code 200 indicates an expired session.

For security reasons, a session should be closed or invalidated once it is not needed anymore. You can achieve this by including *Connection: close* in the header of the final call involving the session in question. As a result, the Cloud Connector invalidates the session. Subsequent attempts to send a request in the context of that session respond with a login page as explained above.

Back to [Top \[page 728\]](#)

User Roles

The REST API supports different user roles. Depending on the role, an API grants or denies access. In default configuration, the Cloud Connector uses local user storage and supports the single user *Administrator* (administrator role). Using LDAP user storage, you can use various users:

Role	Technical Role Name	Authorization
Administrator	admin or sccadmin	All operations.
Support	sccsupport	Edit log and trace configuration.
Display	sccdisplay	Read configuration.
Monitoring	sccmonitoring	Read monitoring information.

Back to [Top \[page 728\]](#)

Return Codes

Successful requests return the code 200, or, if there is no content, 204. POST actions that create new entities return 201, with the location link in the header (that is, the value of the header field *location* is the full URI of the entity created).

The following general errors can be returned by each API:

- 400 – Invalid request. For example, if parameters are invalid or the API is not supported anymore, or an unexpected state occurs, as well as in case of other non-critical errors.
- 401 – Authorization required.
- 403 – The current Cloud Connector instance does not allow the request. Typically, this is the case when the initial password has not been changed yet.
- 404 – The specified entity does not exist.
- 405 – An entity does not support the requested operation.
- 409 – Current state of the Cloud Connector does not allow a particular request as it conflicts with certain rules or violates certain constraints.
- 415 – Unexpected MIME type.
- 500 – Operation failed.

Most APIs return specific error details depending on the situation. Such errors are addressed by the respective API description.

Back to [Top \[page 728\]](#)

Hypertext Application Language

Entities returned by the APIs contain links as suggested by the current draft *JSON Hypertext Application Language* (see <https://tools.ietf.org/html/draft-kelly-json-hal-08> ).

Back to [Top \[page 728\]](#)

Available REST APIs

The Cloud Connector provides APIs for:

[Monitoring tasks \[page 670\]](#)

[Configuration tasks \[page 482\]](#)

Back to [Top \[page 728\]](#)

Sample Code

Some APIs include examples with command line invocations typically involving curl. The use of single and double quotes in those examples depends on the operating system and may have to be adapted.

[Back to Top \[page 728\]](#)

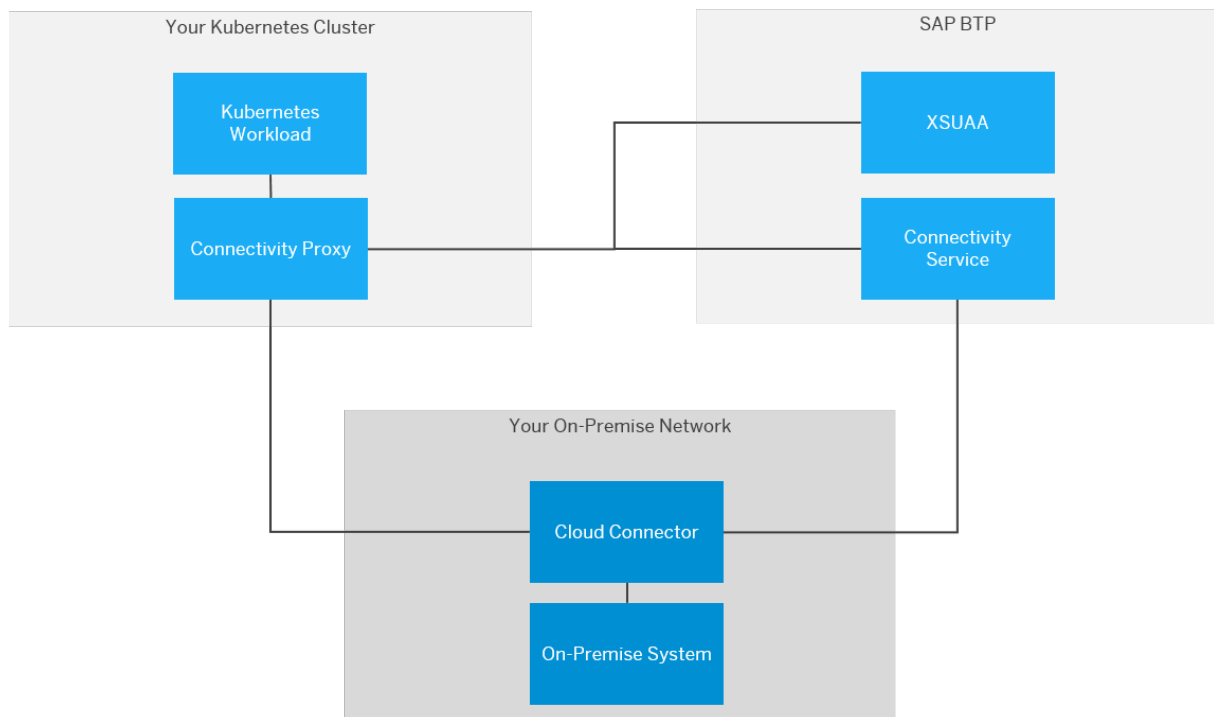
1.3 Connectivity Proxy for Kubernetes

Use the connectivity proxy for Kubernetes to connect workloads on a Kubernetes cluster to on-premise systems.

The connectivity proxy is a Kubernetes component that connects workloads running on a Kubernetes cluster to on-premise systems, which are exposed via the [Cloud Connector \[page 343\]](#). The connectivity proxy must be paired to an SAP BTP region to grant access to the Cloud Connectors connected to that region. The SAP BTP domain model (subaccounts) is used to target a particular Cloud Connector.

The connectivity proxy is delivered as a Docker image and a Helm chart. You need to run the image on your Kubernetes cluster with appropriate configurations. The Helm chart simplifies the installation process. See [Lifecycle Management \[page 764\]](#) for more details.

You can find information about new versions of the connectivity proxy via SAP BTP [release notes](#). To request a new feature, you can use [Influence SAP](#).



Related Information

[Concepts \[page 733\]](#)

[Lifecycle Management \[page 764\]](#)
[Verification and Testing \[page 785\]](#)
[Monitoring \[page 787\]](#)
[Using the Connectivity Proxy \[page 788\]](#)
[Troubleshooting \[page 792\]](#)
[Frequently Asked Questions \[page 799\]](#)

1.3.1 Concepts

Find an overview of important concepts for working with the connectivity proxy for Kubernetes.

How the Connectivity Proxy Works [page 733]	Learn about the connectivity proxy for Kubernetes: Scenario and configuration steps.
Operational Modes [page 737]	Details about the different operational modes of the connectivity proxy.
Mutual TLS [page 741]	Use Transport Layer Security (TLS) with the connectivity proxy.
External Health Checking [page 745]	Perform external health checks for the connectivity proxy.
High Availability [page 749]	Run the connectivity proxy for Kubernetes in high availability mode.
Audit Logging [page 751]	Using audit logging for the connectivity proxy.
Integration with SAP Services [page 752]	Integrate the connectivity proxy with SAP services.
Automatic Pickup on Resource Changes [page 755]	Configure automatic pickup on resource changes for the connectivity proxy.
Service Channels: On-Premise-to-Cloud Connectivity [page 756]	Use the connectivity proxy to connect to an on-premise network via service channel.
Installing the Connectivity Proxy in Clusters with Istio [page 758]	Install the connectivity proxy in a Kubernetes cluster that is configured for Istio.
Installing the Transparent Proxy as Subchart of the Connectivity Proxy [page 760]	Install the transparent proxy as subchart of the connectivity proxy via Helm.
Installing the Connectivity Proxy in Multi-Region Mode [page 760]	Use a single connectivity proxy installation for your global account.

1.3.1.1 How the Connectivity Proxy Works

Learn about the connectivity proxy for Kubernetes: Scenario and required configuration.

[Glossary \[page 734\]](#)
[Scenario \[page 734\]](#)

- [Prerequisites \[page 735\]](#)
- [Runtime Flow in Steps \[page 735\]](#)
- [Involved Parties \[page 736\]](#)

[Configuration \[page 736\]](#)

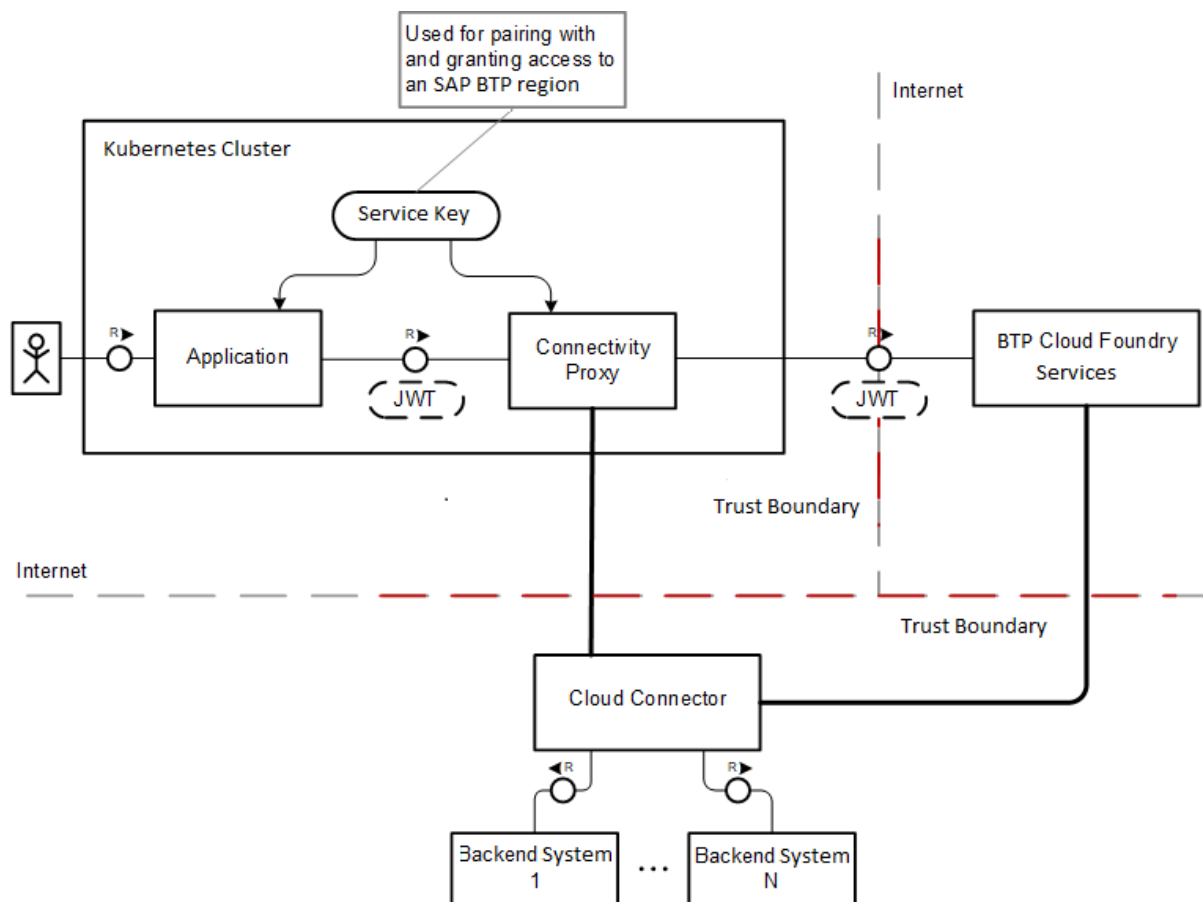
Glossary

- **SAP Business Technology Platform** (SAP BTP): SAP platform for cloud services (replaces *SAP Cloud Platform*).
- **SAP Connectivity service**: Core platform service, offering a secure tunnelling solution between your on-premise network and the cloud.
- **Cloud Connector**: On-premise client of the Connectivity service, deployed and lifecycle-managed in your local network. The initiator of the secure tunnel to the platform, that is, to the Connectivity service.
- **Connectivity proxy**: Software component (logically part of the Connectivity service), deployed locally to the consuming part (usually a cloud application or a service component). It can work in multiple operational modes, depending on the exact requirement of the consuming party.
- **SAP UAA** (aka XSUAA): SAP Authorization service, issues client credentials and access tokens, associated with the platform tenancy model.

[Back to Top \[page 733\]](#)

Scenario

An end user works with a cloud application or solution. To complete the task, the application or solution needs to connect to an on-premise system (hosted either by the consumer tenant or the cloud application provider tenant). The system is not accessible directly via Internet, but securely exposed by the Cloud Connector. Only selected parts of the system functionality may be exposed to the cloud application. For more information, see [Cloud Connector \[page 343\]](#).



Prerequisites

- The connectivity proxy is deployed and configured in the Kubernetes cluster (see [Lifecycle Management \[page 764\]](#)).
- A cloud application is deployed on Kubernetes, next to the connectivity proxy, and it is configured to connect to the proxy (see [Using the Connectivity Proxy \[page 788\]](#)). The cloud application is up and running and accessible by end users.
- The Cloud Connector is installed and configured in your local network and connected to the cloud (that is, to the Connectivity service), and stays in a ready-to-be-used mode (see [Cloud Connector \[page 343\]](#)).
- The on-premise systems that the cloud application needs to connect to are properly exposed via / configured in the Cloud Connector (see [Configure Access Control \[page 456\]](#)).

[Back to Scenario \[page 734\]](#)

[Back to Top \[page 733\]](#)

Runtime Flow in Steps

When all prerequisites are met, the cloud application can be properly used by end users:

- An end user works with a *client tool*, for example, a browser or a REST client.
- The *client tool* connects to a *cloud application*, in this case hosted in a Kubernetes cluster.

- The *cloud application* knows it needs to connect to the on-premise system, therefore it connects to the connectivity proxy. Depending on the specific requirements of the cloud application environment, the *cloud application* might need to obtain an access token which is used to authenticate the application as a client, as well as to authorize it to connect to the respective tenant-specific Cloud Connector exposing the target system (see [Operational Modes \[page 737\]](#)).
- The *connectivity proxy* accepts the client proxy request (by the *cloud application*) and routes the traffic via a TLS secure tunnel, which has been already initiated by the Cloud Connector and successfully established.
- The Cloud Connector receives the *request data* from the *cloud application* and performs the related access control checks. A connection is established to the target on-premise system, and the request data is forwarded to the on-remise system.
- The *response data* from the on-premise system is routed back to the *cloud application*.
- The *cloud application* processes the response data retrieved by the on-premise system and shows the result to the end user.

[Back to Scenario \[page 734\]](#)

[Back to Top \[page 733\]](#)

Involved Parties


- **Cloud application:** Business workload initiated by end users (or a background job) of the business solution.
- **SAP BTP Services:** The connectivity proxy cannot operate on its own. It needs to connect to other services for key operations, namely:
 - *authorization - XSUAA:* Ensure any operation is properly secured.
 - *pairing/integration with SAP Connectivity service:* Secure access control to Cloud Connectors.
- **On-Premise systems or services:** The target system, securely hosted in your local network, usually behind a firewall, and exposed via the Cloud Connector.

[Back to Scenario \[page 734\]](#)

[Back to Top \[page 733\]](#)

Configuration

The connectivity proxy for Kubernetes is meant to

- Be deployed as a Kubernetes [StatefulSet](#) . As it is accessed both by other Kubernetes deployments and the Cloud Connector, you need to configure access accordingly.
- Connect to other (remote) services.
- Enforce authorization on its proxy endpoints.

For all these points, proper configuration is required. For more information, see [Lifecycle Management \[page 764\]](#).

[Back to Top \[page 733\]](#)

1.3.1.2 Operational Modes

Learn about the different operational modes of the connectivity proxy for Kubernetes.

The connectivity proxy can run in four different operational modes, based on two main categories:

- Trust for the surrounding environment and the callers of the proxy
- Tenant usage of the proxy

Find below the details for each operational mode:

[Single-Tenant Usage: *Non-Trusted* Environment \[page 737\]](#)

[Multi-Tenant Usage: *Non-Trusted* Environment \[page 738\]](#)

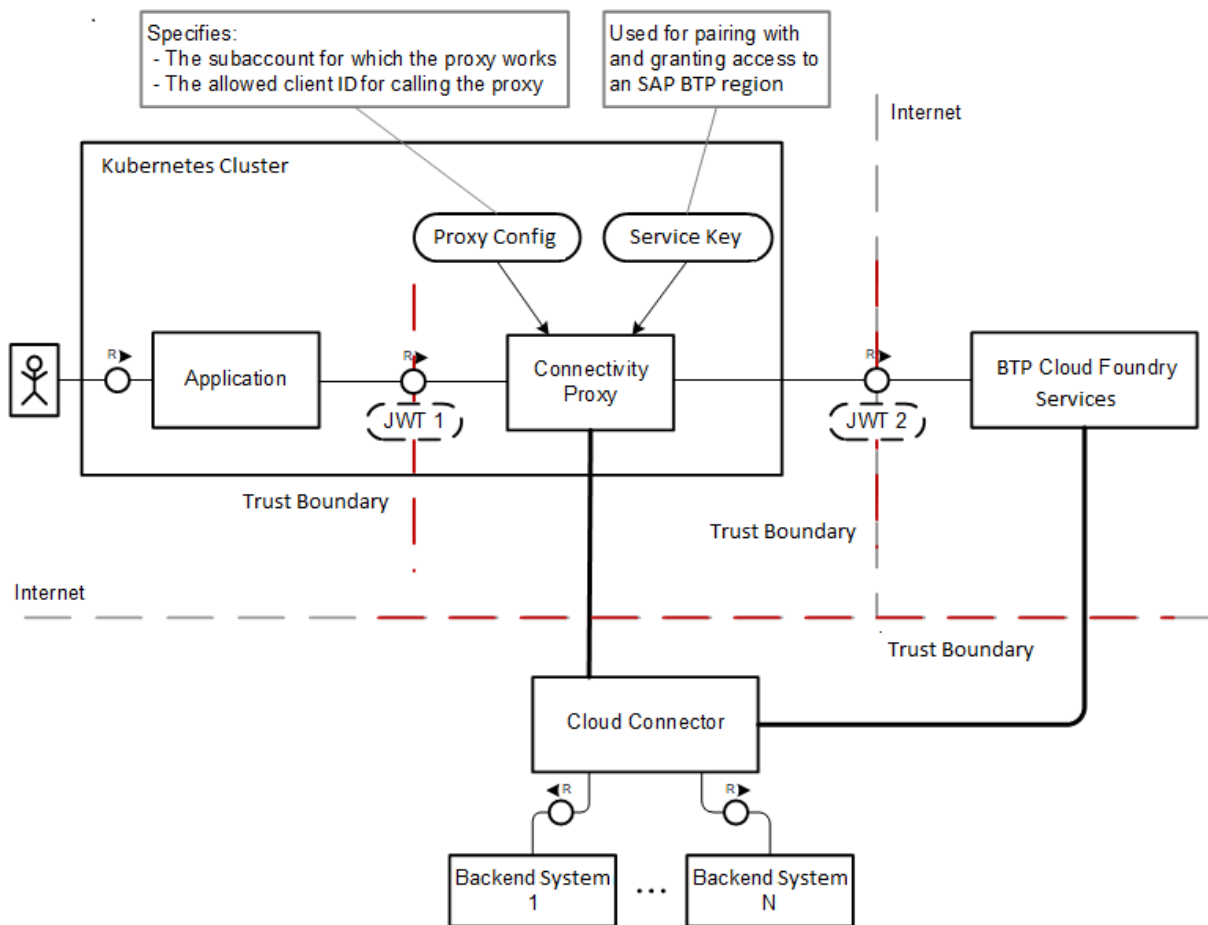
[Single-Tenant Usage: *Trusted* Environment \[page 739\]](#)

[Multi-Tenant Usage: *Trusted* Environment \[page 740\]](#)

Single-Tenant Usage: *Non-Trusted* Environment

The connectivity proxy operates on behalf of a single, statically configured tenant. Applications cannot be trusted. The connectivity proxy is configured as follows:

- Proxy authorization is *enabled*.
- Tenant mode is *dedicated*.
- Uses a statically configured service key of the `connectivity_proxy` service instance of the `connectivity` service.
- Connects to the central Connectivity service on behalf of the statically configured tenant.

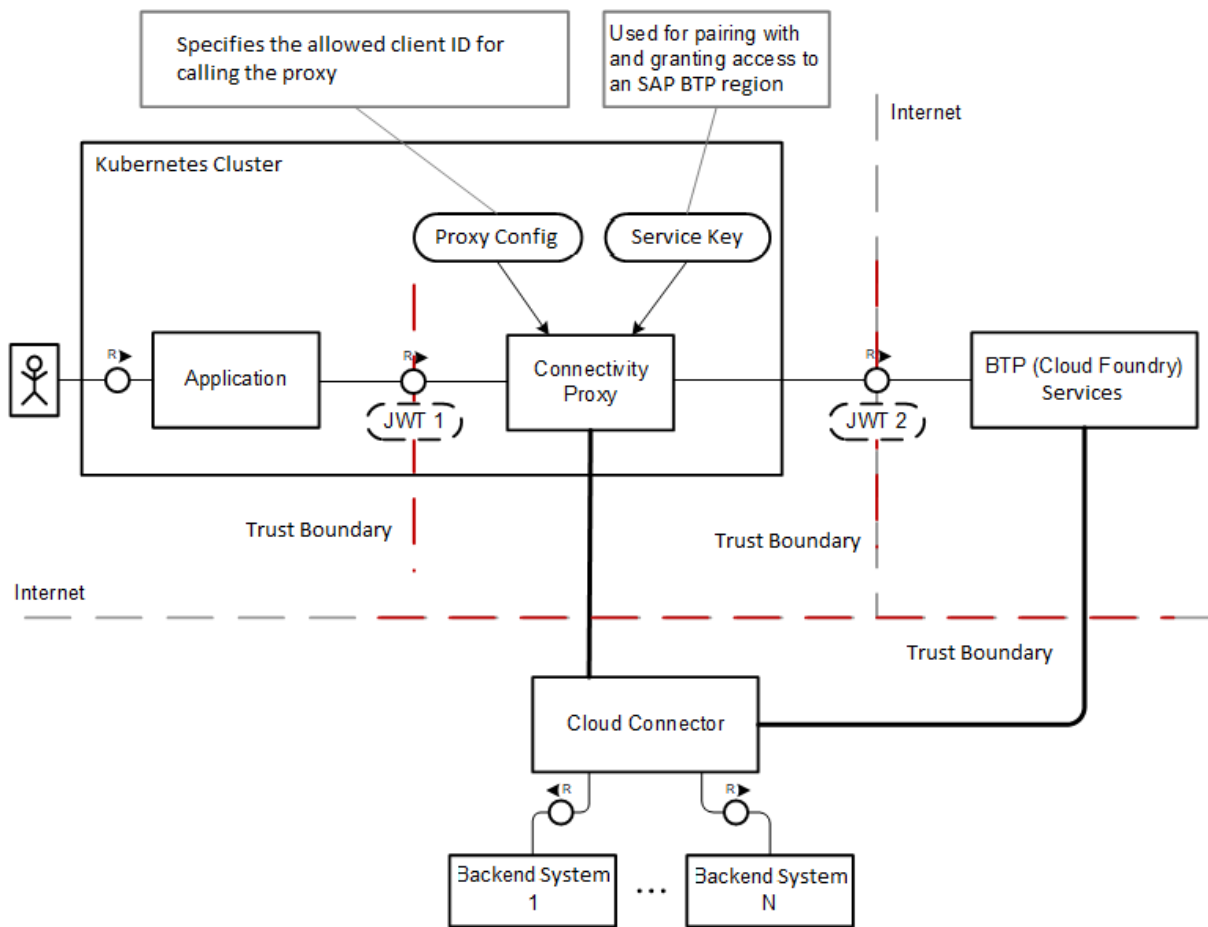


[Back to Top \[page 737\]](#)

Multi-Tenant Usage: *Non-Trusted Environment*

The connectivity proxy operates on behalf of multiple tenants. Applications cannot be trusted. The connectivity proxy is configured as follows:

- Proxy authorization is *enabled*.
- Tenant mode is *shared*.
- Uses a statically configured service key of the `connectivity_proxy` service instance of the `connectivity` service.
- Connects to the central Connectivity service on behalf of the dynamically determined tenant, based on the OAuth access token (JWT) forwarded by the application.

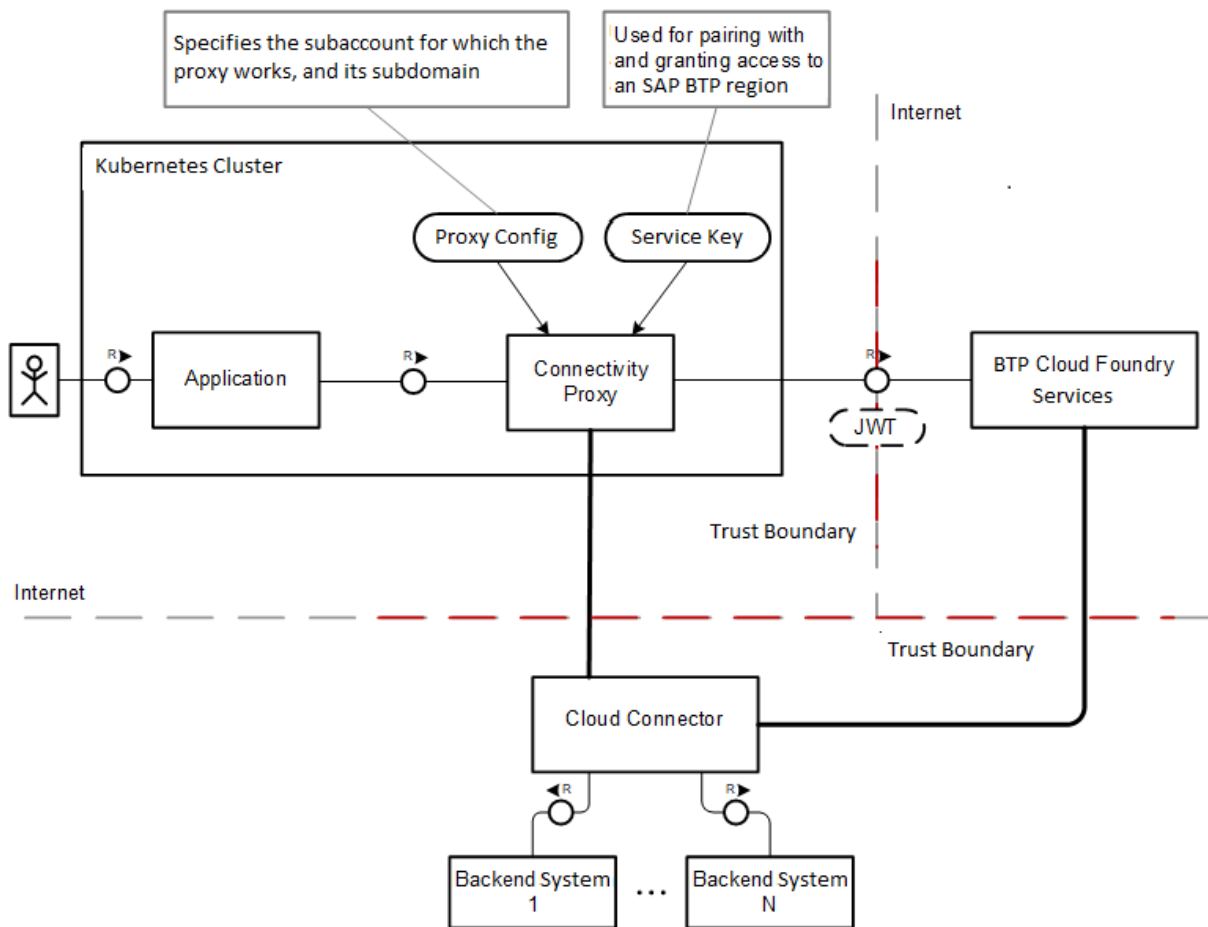


[Back to Top \[page 737\]](#)

Single-Tenant Usage: *Trusted Environment*

The connectivity proxy operates on behalf of a single, statically configured tenant. Applications are trusted. The connectivity proxy is configured as follows:

- Proxy authorization is *disabled*.
- Tenant mode is *dedicated*.
- Uses a statically configured service key of the `connectivity_proxy` service instance of the `connectivity` service.
- Connects to the central Connectivity service on behalf of the statically configured tenant.

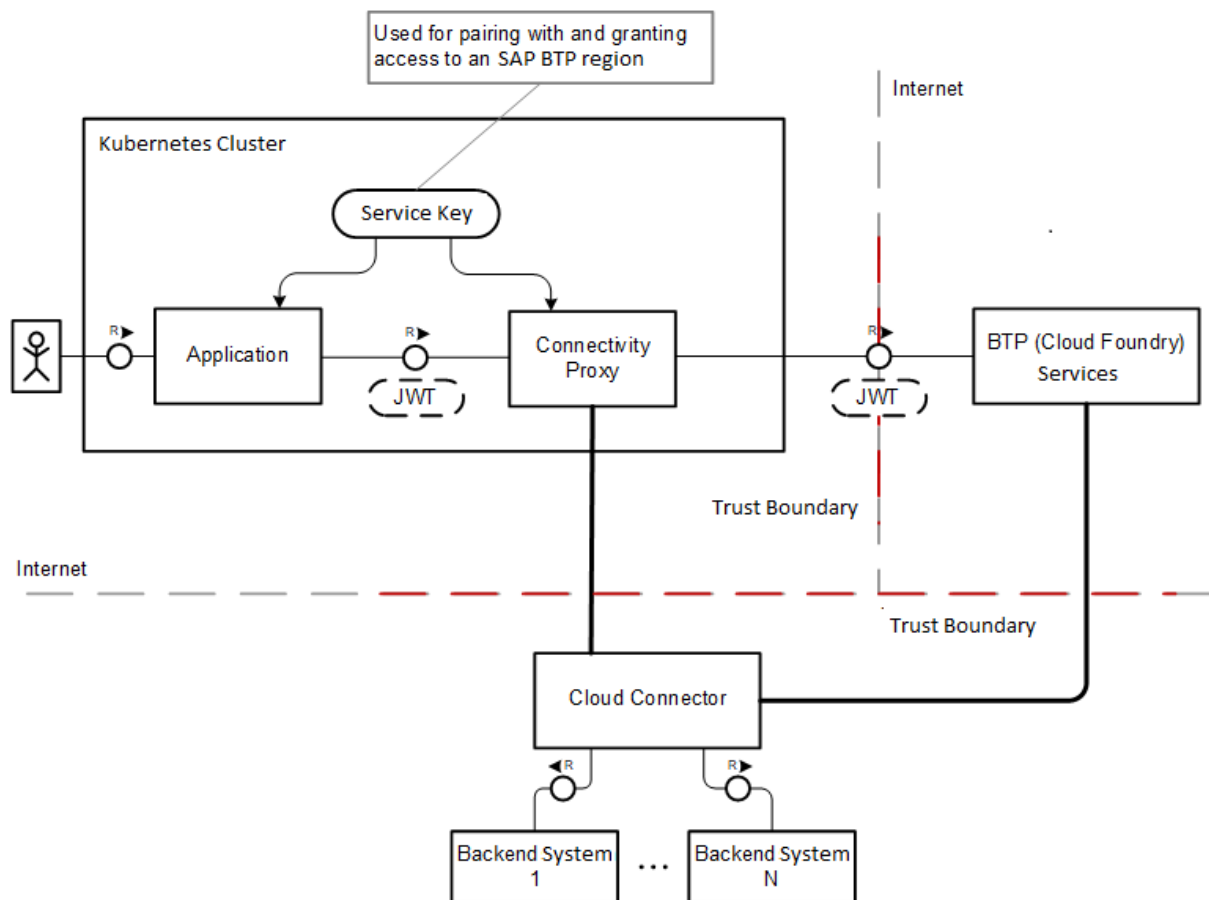


[Back to Top \[page 737\]](#)

Multi-Tenant Usage: *Trusted Environment*

The connectivity proxy operates on behalf of multiple tenants. Applications are trusted. Applications use service keys of the `connectivity_proxy` service instance of the `connectivity` service. The connectivity proxy is configured as follows:

- Proxy authorization is *disabled*.
- Tenant mode is *shared*.
- Connects to the central Connectivity service on behalf of the dynamically determined tenant, using the OAuth access token (JWT) forwarded by the application.



[Back to Top \[page 737\]](#)

Related Information

[Installing the Connectivity Proxy in Multi-Region Mode \[page 760\]](#)

1.3.1.3 Mutual TLS

Use Transport Layer Security (TLS) for the connectivity proxy for Kubernetes.

TLS encrypts the connection between client and server, following the TLS specification. When using mutual TLS, both the TLS client and the TLS server authenticate each other through X.509 certificates.

In an on-premise network, the TLS client is represented by the Cloud Connector. On the cloud side, the direct TLS server may be:

- The Kubernetes Ingress: In this case, the Ingress terminates the TLS connection and establishes a new TLS connection to the connectivity proxy.

- The connectivity proxy: In this case, the Ingress does not terminate the TLS, but transparently forwards the TLS traffic to the connectivity proxy.

Connectivity proxy deployment provides two options to configure end-to-end mutual TLS, that is, the TLS communication between Cloud Connector and connectivity proxy:

- [With TLS termination in the Ingress \[page 742\]](#): TLS configuration has to be made on both the Ingress controller (TLS client) and connectivity proxy (TLS server).
- [Without TLS termination in the Ingress \[page 744\]](#): TLS configuration has to be made only on the connectivity proxy side (TLS server).

End-to-End Mutual TLS *with* Termination of the TLS Connection in the Ingress

Perform the following steps:

1. Enable the TLS communication on the connectivity proxy (TLS server) side, adding the following configuration in the `values.yaml` file:

```
config:
  servers:
    businessDataTunnel:
      enableTls: true
```

2. Add the required TLS configuration for the connectivity proxy (TLS server). There are two options to do this:
 - If you already have an appropriate secret for this purpose in the cluster, add the following configuration in the `values.yaml` file:

```
secretConfig:
  servers:
    businessDataTunnel:
      secretName: <secret name>
```

ⓘ Note

The Kubernetes secret must contain the following properties, used for authentication to the Ingress resource:

- `tls.key`: Base 64-encoded private key in PEM format.
- `tls.crt`: Base 64-encoded certificate in PEM format.
- `ca.crt`: Base 64-encoded full certificate authority (CA) chain in PEM format.

- If you don't have such a Kubernetes secret, it can be automatically generated. Add the following configuration in the `values.yaml` file:

```
secretConfig:
  servers:
    businessDataTunnel:
      secretName: <secret name>
```

```
secretData:
  key: <base 64-encoded private key in PEM format>
  certificate: <base 64-encoded certificate in PEM format>
  caCertificate: <base 64-encoded full Certificate Authority chain
in PEM format>
```

Note

The certificate must be issued for the external host specified on the configuration path `config.servers.businessDataTunnel.externalHost` or for the domain to which the external host belongs. For example, if the external host is `"ingress.mycluster.com"`, the certificate CN or SAN must contain `"ingress.mycluster.com"` or `"*.mycluster.com"`.

3. Add the required TLS configuration for the Ingress. There are three options to do this:
 - If you already have an appropriate Kubernetes secret for this purpose in the cluster, add the following configuration in the `values.yaml` file:

```
ingress:
  tls:
    proxy:
      secretName: <secret name>
```

Note

The Kubernetes secret must contain the following properties, used for authentication to the Ingress resource:

- `tls.key`: Base 64-encoded private key in PEM format.
 - `tls.crt`: Base 64-encoded certificate in PEM format.
 - `ca.crt`: Base 64-encoded full certificate authority (CA) chain in PEM format.
- If you don't have such a secret, it can be automatically generated. Add the following configuration in the `values.yaml` file:

```
ingress:
  tls:
    proxy:
      secretName: <secret name>
      secretData:
        key: <base 64-encoded private key in PEM format>
        certificate: <base 64-encoded certificate in PEM format>
        caCertificate: <base 64-encoded full Certificate Authority chain
in PEM format>
```

- If you didn't add any TLS configuration for the Ingress, the TLS configuration of the connectivity proxy is reused for the Ingress, that is, the connectivity proxy and the Ingress will use the same TLS configuration to communicate to each other.

Note

In this case, the specified certificate must be issued by the specified certificate authority.

Note

The certificate must be issued for the external host specified on the configuration path `config.servers.businessDataTunnel.externalHost` or for the domain to which the external

host belongs. For example, if the external host is "ingress.mycluster.com", the certificate CN or SAN must contain "ingress.mycluster.com" or "*.mycluster.com".

Note

In this case, the **minimum required version** for the NGINX ingress controller is 0.31.0.

[Back to Top \[page 741\]](#)

End-to-End Mutual TLS *without* Termination of the TLS Connection in the Ingress

Perform the following steps:

1. Enable the TLS communication on the connectivity proxy (TLS server) side, adding the following configuration in the `values.yaml` file:

```
config:
  servers:
    businessDataTunnel:
      enableTls: true
```

2. Add the required TLS configuration for the connectivity proxy (TLS server). There are two options to do this:
 - If you already have an appropriate secret for this purpose in the cluster, add the following configuration in the `values.yaml` file:

```
secretConfig:
  servers:
    businessDataTunnel:
      secretName: <secret name>
```

Note

The Kubernetes secret must contain the following properties, used for authentication to the Ingress resource:

- `tls.key`: Base 64-encoded private key in PEM format.
- `tls.crt`: Base 64-encoded certificate in PEM format.

For Connectivity proxy release < 2.4.0, an additional field is required:

- `ca.crt`: Base 64-encoded full **Connectivity** certificate authority (CA) chain in PEM format.

- If you don't have such a secret, it can be automatically generated. Add the following configuration in the `values.yaml` file:

```
secretConfig:
  servers:
```

```
businessDataTunnel:
  secretName: <secret name>
  secretData:
    key: <base 64-encoded private key in PEM format>
    certificate: <base 64-encoded certificate in PEM format>
```

Note

For Connectivity proxy release < 2.4.0 an additional field is required:

```
secretConfig:
  servers:
    businessDataTunnel:
      ...
      secretData:
        ...
        caCertificate: <base 64-encoded full Connectivity Certificate
        Authority chain in PEM format>
```

Note

The certificate must be issued for the external host specified on the configuration path `config.servers.businessDataTunnel.externalHost` or for the domain to which the external host belongs. For example, if the external host is `"ingress.mycluster.com"`, the certificate CN or SAN must contain `"ingress.mycluster.com"` or `"*.mycluster.com"`.

3. Skip the TLS configuration on the Ingress resource, that is, skip the configuration path `ingress.tls` in the `values.yaml` file.

Note

In this case, you must set the configuration parameter `config.servers.businessDataTunnel.strictSniEnabled` to **false**, because this strict SNI configuration is relevant for a TLS termination in the Ingress. In case of direct TLS connection, strict SNI is supported by default.

Note

By default, the NGINX ingress controller does not support communication without termination of the TLS connection. To support such communication, the `<--enable-ssl-passthrough>` flag must be part of the configuration with which the ingress controller is started. For more information, see [Command line arguments](#) (Kubernetes documentation on github).

[Back to Top \[page 741\]](#)

1.3.1.4 External Health Checking

Perform external health checks for the connectivity proxy for Kubernetes.

External health checking lets you check and monitor the health of the connectivity proxy using an external application. To achieve this, you must call one of the following URLs:

- `https://healthcheck.<config.servers.businessDataTunnel.externalHost>/healthcheck`
- `https://healthcheck.<config.servers.businessDataTunnel.externalHost>`, that will redirect the request to the first URL.

The external health checking configuration depends on the mutual TLS configuration. Currently, the connectivity proxy deployment provides three mutual TLS configuration options:

- [Mutual TLS only to the Ingress \[page 746\]](#)
- [End-to-end mutual TLS with termination of the TLS connection in the Ingress \[page 746\]](#)
- [End-to-end mutual TLS without termination of the TLS connection in the Ingress \[page 747\]](#)

For more information, see [Mutual TLS \[page 741\]](#).

Mutual TLS Only to the Ingress or End-to-End Mutual TLS with Termination of the TLS Connection in the Ingress

In these cases, you have two options to configure the external health checking:

- Reuse the TLS secret that is used for communication with the Cloud Connector. In this case, you don't need to do any extra configuration.

Note

The certificate from the TLS secret must be issued for both the external host specified on the configuration path `<config.servers.businessDataTunnel.externalHost>` and the host `healthcheck.<config.servers.businessDataTunnel.externalHost>`. For example, if the external host is `"ingress.mycluster.com"`, the certificate CN could be `"ingress.mycluster.com"` and certificate SAN must contain `"healthcheck.ingress.mycluster.com"`.

- Use a specific TLS secret only for the communication with the external health checking application. In this case, there are two options to do this:
 - If you already have an appropriate secret for this purpose in the cluster, add the following configuration in the `values.yaml` file:

```
ingress:
  healthcheck:
    tls:
      secretName: <secret name>
```

Note

The Kubernetes secret must contain the following properties, used for authentication to the Ingress resource:

- `tls.key`: Base 64-encoded private key in PEM format.
- `tls.crt`: Base 64-encoded certificate in PEM format.

- If you don't have such a Kubernetes secret, it can be automatically generated. Add the following configuration in the `values.yaml` file:

```
ingress:
  healthcheck:
    tls:
      secretName: <secret name>
      secretData:
        key: <base 64 encoded private key in PEM format>
        certificate: <base 64 encoded certificate in PEM format>
```

Note

The certificate from the TLS secret must be issued for the host `healthcheck.<config.servers.businessDataTunnel.externalHost>`. For example, if the external host is `"ingress.mycluster.com"`, then the certificate CN or SAN must contain `"healthcheck.ingress.mycluster.com"`.

Note

The certificate chain of the CA that issues the certificate from the TLS secret must be imported to the trust store of the external health checking application.

[Back to Top \[page 745\]](#)

End-to-end Mutual TLS *without* Termination of the TLS Connection in the Ingress

In this case, execute the following steps:

1. Download the `plugins` archive and unzip it (for more information, see [Lifecycle Management \[page 764\]](#)).
2. Navigate to the `plugins` folder and install `connectivity-certificate-plugin` by executing the following command:

```
helm plugin install connectivity-certificate-plugin
```

3. Generate a TLS secret by executing the following command:

```
helm get-connectivity-certificates <region_domain> <subaccount> <user>
<password> <namespace> <secret_name>
```

Note

- `region_domain`: BTP region to which you are pairing the connectivity proxy.
- `subaccount`: BTP subaccount, on whose behalf the connectivity proxy is running.
- `user`: BTP subaccount user.

- `password`: Password of the BTP subaccount user.
- `namespace`: Kubernetes namespace to which the secret is created.
- `secret_name`: Name of the Kubernetes secret to be created.

4. Specify a TLS secret that will be used for the communication with the external health checking application. There are two options to do this:

- If you already have an appropriate secret for this purpose in the cluster, add the following configuration in the `values.yaml` file:

```
ingress:
  healthcheck:
    tls:
      secretName: <secret name>
```

Note

The Kubernetes secret must contain the following properties, used for authentication to the Ingress resource:

- `tls.key`: Base 64-encoded private key in PEM format.
- `tls.crt`: Base 64-encoded certificate in PEM format.

- If you don't have such a Kubernetes secret, it can be automatically generated. Add the following configuration in the `values.yaml` file:

```
ingress:
  healthcheck:
    tls:
      secretName: <secret name>
      secretData:
        key: <base 64 encoded private key in PEM format>
        certificate: <base 64 encoded certificate in PEM format>
```

Note

The certificate from the TLS secret must be issued for the host `healthcheck.<config.servers.businessDataTunnel.externalHost>`. For example, if the external host is `"ingress.mycluster.com"`, then the certificate CN or SAN must contain `"healthcheck.ingress.mycluster.com"`.

Note

The certificate chain of the CA that issues the certificate from the TLS secret must be imported to the trust store of the external health checking application.

5. Specify the TLS secret that was generated in step 3, to be used for the communication between the Ingress controller and the connectivity proxy. Add the following configuration in the `values.yaml` file:

```
ingress:
  healthcheck:
    tls:
      proxy:
        secretName: <name of the secret generated on step 3>
```

[Back to Top \[page 745\]](#)

1.3.1.5 High Availability

Run the connectivity proxy for Kubernetes in a high-availability setup.

Content

[Overview \[page 749\]](#)

[High-Availability Modes \[page 750\]](#)

[Advantages and Drawbacks \[page 750\]](#)

[High-Availability Mode: *Path* \[page 750\]](#)

[High-Availability Mode: *Subdomain* \[page 751\]](#)

Overview

The connectivity proxy can work in an active-active high-availability setup. In this setup, there are at least two connectivity proxy instances, both running actively and simultaneously.

The main purpose of an active-active deployment is to provide high availability and allow zero-downtime maintenance as well as horizontal-scaling capabilities.

The Kubernetes service exposing the connectivity proxy pods distributes and load-balances the traffic from the workloads across all running connectivity proxy pods.

📘 Note

The load balancing strategy for distributing traffic to the connectivity proxy pods depends on the *kube-proxy* mode. The strategies used by the different *kube-proxy* modes are described in the [Kubernetes documentation](#) 📖. This configuration is done on cluster level.

→ Tip

[MultiAZ with Pod Anti-Affinity](#) 📖 and [PodDisruptionBudget](#) 📖 are supported out of the box in the Helm chart.

[Back to Content \[page 749\]](#)

High-Availability Modes

There are two technical options (modes) for the active-active deployment:

- *Path*
- *Subdomain*

The difference between the two modes is in the way the routing information of the current connectivity proxy (which is being used for the current connection) is passed to the Cloud Connector.

Depending on the perspective, as well as on the concrete requirements and boundaries you have, both modes may have advantages and drawbacks, so you should carefully choose the one which best suits your needs.

The number of connectivity proxy instances that run simultaneously depends on the *replicaCount* configuration in the `values.yml` file.

```
deployment:  
  replicaCount: <number_of_replicas>
```

[Back to Content \[page 749\]](#)

Advantages and Drawbacks

High-Availability Mode	Path	Subdomain
Compatibility with end-to-end mutual TLS without termination	No	Yes
Compatibility with secure and inexpensive certificates	Yes	No

[Back to Content \[page 749\]](#)

High-Availability Mode: *Path*

To configure the connectivity proxy for the high-availability mode *Path*, add the following configuration in the `values.yml` file:

```
config:  
  highAvailabilityMode: "path"
```

[Back to Content \[page 749\]](#)

High-Availability Mode: *Subdomain*

To configure the connectivity proxy for the high-availability mode *Subdomain*, add the following configuration in the `values.yml` file:

```
config:  
  highAvailabilityMode: "subdomain"
```

Note

High-Availability mode *Subdomain* uses either wildcard certificates or SAN (Subject Alternative Name) certificates. Wildcard certificates cover all possible subdomains. However, they are insecure since when the certificate key that is installed on one subdomain gets compromised, it will be compromised on all of the subdomains it is installed on. A safer option is to use SAN certificates, although they may be expensive and require each subdomain to be specified explicitly.

[Back to Content \[page 749\]](#)

1.3.1.6 Audit Logging

Use audit logging for the connectivity proxy for Kubernetes.

At runtime, the connectivity proxy produces audit messages for the relevant auditable events. These audit messages help you gather audit data which can be useful for identifying possible attacks or unwanted access attempts. The audit messages are written on the standard output and it is up to the operator of the connectivity proxy to configure the surrounding infrastructure in a way that these audit messages are collected, stored, protected from tampering, rotated when necessary, and so on.

Note

When written to the standard system output, the audit log messages are preceded by the prefix `[AUDIT-EVENT]` to distinguish them from the other logs.

Audit Messages

In the table below, the left column shows a description of the security event, the right column shows the corresponding audit message. These messages are written on behalf of the consumer tenant.

Note

The `<message>` field of the audit message should be treated as case insensitive.

Operation	Audit Message
<p>[Security Event]</p> <p>Successfully opened tunnel</p> <p>from Cloud Connector to connectivity proxy.</p>	<pre>{"user": "YYY", "data": {"action": "open connectivity tunnel", "message": "tunnel handshake success", "tunnelId": "NNN", "objectName": "connectivity tunnel management", "clientId": "FFF", "loggedBy Class": "com.sap.core.connectivity.tunnel.server.TunnelServerHandshaker", "handshakeDetails": "success"}}, {"tenant": "TTT"}</pre>
<p>[Security Event]</p> <p>Failed attempt to open tunnel</p> <p>from Cloud Connector to connectivity proxy.</p>	<pre>{"user": "YYY", "data": {"action": "open connectivity tunnel", "message": "tunnel handshake failure", "tunnelId": "NNN", "objectName": "connectivity tunnel management", "clientId": "FFF", "loggedBy Class": "com.sap.core.connectivity.tunnel.server.TunnelServerHandshaker", "handshakeDetails": "UUU"}}, {"tenant": "TTT"}</pre>

In the table above, there are multiple keys with UPPER CASE values. At runtime, these keys hold real scenario values:

- YYY: User executing the operation, if known. If the request is not done on behalf of a user, the value will be `fallback_for_missing_user`.
- NNN: Secure tunnel between the connectivity proxy and the Cloud Connector.
- TTT: Tenant on whose behalf the operation is executed.
- FFF: Remote client (unique identifier of a particular installation) of the Cloud Connector.
- UUU: Additional information about the event.

1.3.1.7 Integration with SAP Services

Integrate the connectivity proxy for Kubernetes with SAP services.

[Connectivity Service \[page 753\]](#)

Use the connectivity proxy for Kubernetes with the Connectivity service to connect to on-premise systems.

[Security \(Proxy Authorization\) \[page 753\]](#)

Perform proxy authorization for the connectivity proxy.

1.3.1.7.1 Connectivity Service

Use the connectivity proxy for Kubernetes with the Connectivity service to connect to on-premise systems.

On-premise systems are exposed via the Cloud Connector. The Cloud Connector is connected to (registered in) the SAP Connectivity service and ready to serve. The connectivity proxy is the software component to which, at runtime, the Cloud Connector connects and establishes the business data tunnel.

To use the connectivity proxy, it must be paired with the SAP Connectivity service, that is, configured to connect to the central service to which initially the Cloud Connector was connected. This enables the connectivity proxy to dynamically (on-demand) serve its purpose, that is, to securely establish business data tunnel connections to the Cloud Connector, hosted in an on-premise network next to the backend systems.

Prerequisites

You have an account on SAP BTP, Cloud Foundry environment, and your SAP BTP account user has the authorization to create service instances.

Procedure

1. Create a Connectivity service instance with service plan `connectivity_proxy`.
2. Create a service key of the created service instance.

→ Tip

You can create the service key with x.509 credentials instead of a client secret, but only in the SAP-managed certificate case. If you do so, you must take care to rotate the key before the certificate expires.

3. Follow the instructions under [Lifecycle Management \[page 764\]](#) and set up the Kubernetes secret for pairing with the central SAP Connectivity service, providing the content of the created service key.
4. Check the [Configuration Guide \[page 772\]](#) for `config.integration.connectivityService*` parameters.

1.3.1.7.2 Security (Proxy Authorization)

Perform proxy authorization for the connectivity proxy for Kubernetes.

An important aspect of using the connectivity proxy is the proxy authorization, that is, how the access to the proxy is controlled (protected), enabling the callers to reach out to on-premise systems exposed via the Cloud Connector.

As described in [Operational Modes \[page 737\]](#), there are two major categories for consideration: The level of trust in the environment, and the *type of tenancy* (single vs multi-tenant). This helps you prevent unwanted callers and is especially useful in scenarios where calls to the proxy can originate from workloads outside of your control, or if you simply want to apply stricter rules.

The authorization of the connectivity proxy is based on OAuth and it is provided via integration with the SAP UAA (aka XSUAA) service acting as an OAuth server. For a successful authorization request, a JSON Web token (JWT) must be sent to the connectivity proxy, for which the following is valid:

- Issued by XSUAA
- Not expired
- Passes the signature verification (the connectivity proxy takes care to get the token keys from XSUAA for offline JWT verification).
- Its `client_id` (the client ID of the OAuth client which is part of the respective XSUAA service instance credentials) matches the allowed client ID in the connectivity proxy configuration.

```
config:
  servers:
    proxy:
      authorization:
        oauth:
          allowedClientId: "the client id which is allowed as JWT issuer for proxy authorization"
```

→ Tip

The service instance used to protect the connectivity proxy can be any XSUAA service instance.

📌 Note

Based on the above info, the connectivity proxy can be protected with the very same service instance you use to perform the user login flow for your application. In this case, you can reuse the login token. However, to achieve separation of concerns, we recommend that you use a dedicated service instance to protect the connectivity proxy.

Multi-Tenancy and Subscriptions

An important aspect of the proxy authorization is that, when enabled, it is also used to pass the tenant context for the request to the connectivity proxy. When you fetch a token for proxy authorization, make sure you do so on behalf of the *correct* tenant, over which you want to call the connectivity proxy.

In cases of multi-tenancy, or single-tenancy where the proxy-protecting instance is created in a tenant that is *different* from the one for which the proxy is dedicated, you must *ensure subscriptions* between this tenant and the proxy-protecting instance to fetch a token on behalf of the correct tenant.

1.3.1.8 Automatic Pickup on Resource Changes

Configure automatic pickup on resource changes for the connectivity proxy.

The connectivity proxy can automatically pick up changes in secrets and configmaps. This is achieved by performing a rolling restart when a resource is modified.

Restart Watcher

The restart watcher is responsible for picking up changes in the above-mentioned resources, and for restarting the connectivity proxy if given conditions are met.

It will be informed for changes in a secret or configmap containing the label `connectivityproxy.sap.com/restart`.

To enable the restart watcher, you must configure the following in the `values.yml` file.

```
deployment:
  restartWatcher:
    enabled: true
```

Note

The restart watcher is unique for every Helm installation of the connectivity proxy and will be deployed as a part of it.

Caution

If the connectivity proxy:

- is with initial installation version < 2.5.0
- is upgraded via Helm to version 2.5.0 or higher
- has the restart watcher enabled
- and has replica count 2

the restart operation which is performed by the restart watcher may not be a ZDM (zero-downtime migration).

To avoid such situation, you should perform a fresh Helm installation. This is inevitable because a change in an immutable field of a Kubernetes resource is required.

Conditions for Restart

The restart watcher checks for two possible values of the label `connectivityproxy.sap.com/restart`:

- if the value is an empty string, the restart watcher will perform a rolling update on the connectivity proxy, based on respective Helm installation.
- if the value has the form `<helm_installation_name>.<namespace>`, the restart watcher will perform a rolling restart only if the values of `<helm_installation_name>` and `<namespace>` match the watcher's installation and namespace.
- for all other possible values, the restart watcher will not perform a restart.

Examples

- You have a Helm installation in the `default` namespace and a secret with label `connectivityproxy.sap.com/restart: ""` (with value empty string) in namespace `test`. In this case, the connectivity proxy installation in namespace `default` will be restarted for every change of the secret.
- You have a Helm installation with name `connectivity-proxy` in namespace `test`, and a configmap with name `test-configmap` and label `connectivityproxy.sap.com/restart : connectivity-proxy.test`. In this case, the connectivity proxy will be restarted for every change of the configmap. However, if you have another installation in the same namespace `test` but with name `connectivity-proxy2`, restarts will not be triggered on it based on changes of the configmap `test-configmap`.

1.3.1.9 Service Channels: On-Premise-to-Cloud Connectivity

Use the connectivity proxy to connect to an on-premise network via a Cloud Connector service channel.

If you want to consume a cloud application running in a Kubernetes cluster, and the application is not accessible via Internet, it can be securely exposed by the connectivity proxy and consumed from the on-premise network via a Cloud Connector service channel.

For more information on configuring service channels in the Cloud Connector, see [Using Service Channels \[page 608\]](#).

Perform the following steps to expose an application through the connectivity proxy:

1. Install the connectivity proxy in the cluster where the cloud application is running.
2. Create a Kubernetes resource of type `ServiceMapping` for every cloud application that you want to expose through the connectivity proxy.

Sample Code

```
apiVersion: connectivityproxy.sap.com/v1
kind: ServiceMapping
```

```
metadata:
  name: <required>
spec:
  type: <required>
  subaccountId: <required>
  serviceId: <required>
  tenantId: <optional>
  internalAddress: <required>
  locationIds: <optional>
```

- `name`: Required. Name of the `ServiceMapping` that is created.
- `type`: Required. Type of the `ServiceMapping`. The type defines the connection protocol used to expose the cloud application. Currently, the supported types are **RFC** and **TCP**.

Note

For Cloud Connectors below version 2.14.2, only **RFC** is supported.

- `subaccountId`: Required for multi-tenant modes and optional for single-tenant modes. This is the ID of the tenant on whose behalf the cloud application is exposed. In single-tenant modes, the configured dedicated tenant is used by default.
- `serviceId`: Required. This is the virtual host used to expose the cloud application.
- `tenantId`: Optional. If the connectivity proxy consumer's flow has a tenancy domain model different from the SAP BTP tenancy domain model, you can specify the tenant identifier of the custom domain model here.
- `internalAddress`: Required. This field is composed of the host and port for the exposed application. The connectivity proxy will use the internal address for opening a connection to the application. This usually is the DNS name of a Kubernetes service and a port.
- `locationIds`: Optional. List of location identifiers for which service channels can be opened. By default, if the list of location identifiers is not provided, only the default location identifier is supported. If more identifiers should be supported and the default one is amongst them, you can add them as "" (empty string) in the list.

For more information about location identifiers, see [Set up Connection Parameters and HTTPS Proxy](#) (step 4).

Example (Service Mapping):

Sample Code

```
spec:
  locationIds:
    - "locationId1"
    - "locationId2"
    - ""
```

Note

The combination of `type`, `subaccountId` and `serviceId` for a service mapping resource must be unique.

Note

The combination of `type`, `subaccountId`, and `serviceId` for a `ServiceMapping` resource should be unique.

3. Once the `ServiceMapping` resource is successfully created, it is picked up and loaded in the connectivity proxy.

⚠ Caution

The `ServiceMapping` is a cluster-scoped Kubernetes resource. That is, the created `ServiceMappings` are not bound to a specific namespace. They will be picked up from all namespaces of the Kubernetes cluster and loaded in the connectivity proxy.

→ Remember

Picking up and loading a newly created `ServiceMapping` resource in the connectivity proxy may take a few seconds, as it is done by Kubernetes-native mechanisms.

4. After the `ServiceMapping` is loaded in the connectivity proxy, the cloud application can be consumed via the Cloud Connector as of version 2.15.0.
5. To stop exposing the cloud application through the connectivity proxy, delete the corresponding `ServiceMapping` resource from the cluster.

1.3.1.10 Installing the Connectivity Proxy in Clusters with Istio

Install the connectivity proxy in a Kubernetes cluster that is configured for Istio.

⚠ Restriction

Currently, in clusters with Istio configured, only the high availability mode "path" is supported. Also, you cannot configure end-to-end mutual TLS (mTLS).

By default, the traffic between Cloud Connector and the ingress gateway is secured by mTLS. The traffic between the ingress gateway pod and the connectivity proxy pod is also encrypted by Istio by default.

Required Configuration

1. The ingress `className` must be changed to "istio". You can specify this in the `values.yaml` file:

📄 Sample Code

```
ingress:
  className: "istio"
```

⚠ Caution

The default value of `ingress.className` is "nginx". It is mandatory to change it if you have Istio configured.

2. By default, Istio is usually installed in the namespace "istio-system". However, you can install it in a different namespace. The connectivity proxy needs to know Istio's installation namespace because it creates secrets used by the Istio ingress gateway when performing the TLS handshake with the Cloud Connector.

The connectivity proxy will create those secrets in the default namespace "istio-system". However, if you have installed Istio in a different namespace, you can specify this in the `values.yaml` file:

🔗 Sample Code

```
ingress:
  istio:
    namespace: <istio-installation-namespace>
```

⚠ Caution

The secret that will be used for TLS configuration of the ingress-gateway should be created in the installation namespace of Istio. This secret is specified with the field `ingress.tls.secretName` in the `values.yaml`. For more information, see [Configuration Guide \[page 772\]](#).

3. The connectivity proxy uses the selector which is configured in the Istio ingress gateway to apply a gateway configuration on the gateway pods. When installing Istio with default configurations, it uses "istio:ingressgateway" as default value for the selector.

The connectivity proxy also has this value configured as default one. If Istio was installed with different configurations and you need to change the selector, you can specify this in the `values.yaml` file:

🔗 Sample Code

```
ingress:
  istio:
    gateway:
      selector:
        "key1" : "value1"
        "key2" : "value2"
```

⚠ Caution

Istio AuthorizationPolicy

If you have configured an Istio authorization policy that could affect traffic to the connectivity proxy, you may need to add the following rules (or the whole authorization policy):

🔗 Sample Code

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: <authorization-policy-name>
```

```
namespace: <istio-installation-namespace>
spec:
  selector:
    matchLabels:
      app: connectivity-proxy
  action: ALLOW
  rules:
  - to:
    - operation:
      ports: ["8042", "20001", "20003", "20004"]
```

1.3.1.11 Installing the Transparent Proxy as Subchart of the Connectivity Proxy

If you install the connectivity proxy with Helm, you can have the transparent proxy installed as a subchart. For more information about Helm subcharts, see the [Helm documentation](#) .

1. The transparent proxy subchart is disabled by default. To enable it, set the `enabled` flag to `true` in the `transparent-proxy` section in the `values.yaml` file.

Sample Code

```
transparent-proxy:
  enabled: true
```

2. You can modify the transparent proxy version from the `requirements.yaml` file in the connectivity proxy release.

Related Information

[Transparent Proxy for Kubernetes \[page 800\]](#)

[Configuration Guide \[page 772\]](#)

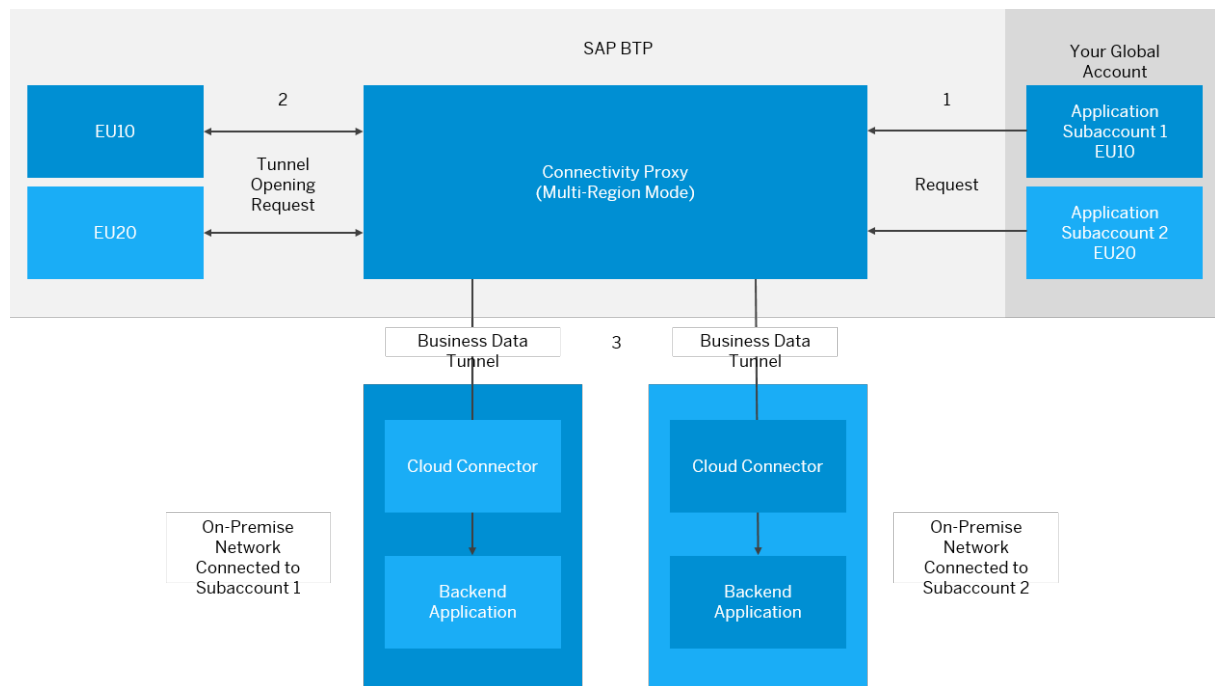
1.3.1.12 Installing the Connectivity Proxy in Multi-Region Mode

Use a single connectivity proxy installation for your global account.

Overview

The connectivity proxy can operate in multi-region mode.

SAP BTP regions represent locations (of data centers) like eu10, eu20, us10, us20, and so on. Using the multi-region mode, a single connectivity proxy installation can operate in different regions which, however, must be part of the same landscape (global account). Otherwise, the uniqueness of subaccount IDs is not guaranteed.



Why is the Multi-Region Mode Valuable?

Using the multi-region mode, you can install *a single connectivity proxy* for your global account. Applications - deployed in different regions or in different subaccounts within the same region - can consume the Connectivity service through this single connectivity proxy instance.

Without multi-region mode, you must install *one connectivity proxy for every region* where applications on cloud side need to consume the Connectivity service. This setup requires considerably more maintenance effort and consumes a lot more resources than just a single (multi-region) instance.

Install the Connectivity Proxy in Multi-Region Mode

1. *Without multi-region mode*, the service keys for integration with other services are provided through the `values.yaml` file in the following way:

```
secretConfig:
```



```

integration:
  connectivityService:
    secretData: <base64 encoded service key>

```

In **multi-region mode**, you can assign different regions to a single connectivity proxy instance. Consequently, the service keys for each region are needed. In this case, you cannot configure the keys through the *values.yaml* file. Instead, you must put them into secrets, and then specify them in a *ConfigMap*.

Let's create a *ConfigMap* containing two region configurations.

First, we configure the two secrets holding the service keys for each region:

```

apiVersion: v1
kind: Secret
metadata:
  name: <secret-name-region1>
data:
  serviceKey: <base64 encoded service key for region1>

```

```

apiVersion: v1
kind: Secret
metadata:
  name: <secret-name-region2>
data:
  serviceKey: <base64 encoded service key for region2>

```

⚠ Caution

The `serviceCredentialsKey` property for each service type specified under `config.integration` must be used in all secrets that will hold service keys for the respective service type. For example, if the following `serviceCredentialsKey` is configured for the Connectivity service in the *values.yaml*...

```

config:
  integration:
    connectivityService:
      serviceCredentialsKey: serviceKey

```

...then it must be used in all secrets holding service keys for the Connectivity service.

In a second step, we create a *ConfigMap* that contains two region configuration IDs (*region1* and *region2*), which may have any value. Each region configuration contains the Connectivity service as dependency, and the secret name holding the specified service key.

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: regionConfigurations
data:
  region1: "{\"dependencies\":{\"connectivity\": \"<secret-name-region1>\"}}"
  region2: "{\"dependencies\":{\"connectivity\": \"<secret-name-region2>\"}}"

```

📌 Note

Region configuration IDs do not require any specific naming pattern.

2. To enable the multi-region mode, add the following in the *values.yaml* file and specify the name of the *ConfigMap* containing the region configurations:

```
config:
  tenantMode: shared
  multiRegionMode:
    enabled: true
    configMapName: "regionConfigurations"
```

Note

When multi-region mode is enabled, the connectivity proxy operates in multi-tenant mode. Therefore, you must set the `tenantMode` property to `shared`.

Caution

If the connectivity proxy is installed in [non-trusted mode \[page 737\]](#) (or proxy authorization is enabled), the [allowed client IDs \[page 753\]](#) for all region configurations must be specified in the *ConfigMap*:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: regionConfigurations
data:
  region1: "{\"allowedClientId\": \"<allowed-client-id>\", \"dependencies\": {\"connectivity\": \"<secret-name>\"}}\"
  region2: "{\"allowedClientId\": \"<allowed-client-id>\", \"dependencies\": {\"connectivity\": \"<secret-name>\"}}\"
```

Caution

The connectivity proxy cannot be installed if both **multi-region mode** and [service channels \[page 756\]](#) are enabled.

Consume the Connectivity Proxy in Multi-Region Mode

If multi-region mode is enabled, you must send the header `SAP-Connectivity-Region-Configuration-Id` with every request made to the connectivity proxy.

Remember

Keep in mind that the correct token for each region configuration must be sent. Sending an access token/ authorization token to the proxy is required with every request, because the connectivity proxy is working in multi-tenant mode.

An example for a request with curl through the HTTP protocol:

Sample Code

```
curl -x connectivity-proxy:20003 "http://<virtual_host>:<virtual_port>/<path>" --header "SAP-Connectivity-Region-Configuration-Id:"
```

```
region1" --header "SAP-CP-Connectivity-Service-Token:
<base64_encoded_jwt_for_the_subaccount_in_the_specified_region_configuration> "
```

In this case, the request is for a region configuration with ID *region1* in the *ConfigMap*.

Note

The access token must be issued for the subaccount that corresponds to the region configuration, identified by the region ID that has been sent with the request.

Caution

If the proxy is installed in non-trusted mode, the authorization token must be provided via the `Proxy-Authorization` header.

Example:

Sample Code

```
curl -x connectivity-proxy:20003 "http://<virtual_host>:<virtual_port>/
<path>" --header "SAP-Connectivity-Region-Configuration-Id: region1" --
header "Proxy-Authorization: Bearer
<base64_encoded_jwt_for_the_subaccount_in_the_specified_region_configuratio
n>"
```

Change Region Configuration

As an operator of the connectivity proxy, at some point you might need to make a change in the region configurations *ConfigMap* or in any of the secrets containing service credentials. For example, add or remove a region configuration, or change the service credentials in a secret. This can be done any time, and no further actions are needed to make the connectivity proxy start working with the performed changes:

- After a short period of time, the changes will be picked up automatically by the connectivity proxy.
- A potentially required restart is also done automatically.
- The restart of the connectivity proxy's pods is rolling.
- If high availability mode is enabled, no downtime is expected.

1.3.2 Lifecycle Management

Use the connectivity proxy image and the connectivity proxy Helm chart to manage the life cycle of the connectivity proxy for Kubernetes.

The connectivity proxy delivery includes the following main components:

- **Connectivity Proxy Image** [page 765]: the functional component that packages all the binaries and utilities.
- **Connectivity Proxy Helm Chart** [page 766]: used for configuring and managing the life cycle of the connectivity proxy via the popular Helm package manager. For more information, see [Operations via Helm](#) [page 767].

Connectivity Proxy Image

The connectivity proxy image is a standard Docker image containing all the required binaries for the connectivity proxy. This includes the connectivity proxy binaries themselves, a JVM (SapMachine), and other important utilities.

Connectivity Proxy Image in DockerHub

The connectivity proxy image is available via the DockerHub image repository, see <https://hub.docker.com/r/sapse/connectivity-proxy> .

Pull information (for the latest version):

- **Registry:** docker.io
- **Repository:** sapse/connectivity-proxy
- **Tag:** 2.12.1

Example:

Sample Code

```
docker pull docker.io/sapse/connectivity-proxy:2.12.1
```

Connectivity Proxy Image in the Repository-Based Shipment Channel (RBSC)

The connectivity proxy image is available via the RBSC docker image repository. This requires having an S-user with associated licenses.

Note

Contact your account manager to clarify access to the repository.

Pull information (for the latest version):

- **Registry:** 73554900100900005672.dockersrv.cdn.repositories.cloud.sap
- **Repository:** com.sap.cloud.connectivity/connectivity-proxy
- **Tag:** 2.12.1

- **Authorization:** see [Manage Technical Users in SAP Repositories Management](#) (RBSC documentation).

Example:

```
docker pull 73554900100900005672.dockersrv.cdn.repositories.cloud.sap/
com.sap.cloud.connectivity/connectivity-proxy:2.12.1
```

[Back to Top \[page 2\]](#)

Connectivity Proxy Helm Chart

The connectivity proxy delivery also includes a Helm chart that you can use for life cycle management. It is the recommended way to perform life cycle management. The connectivity proxy Helm provides full configuration capabilities via the standard Helm method of a `values.yaml` file (see [Configuration Guide \[page 772\]](#)).

Connectivity Proxy Helm Chart in the Repository-Based Shipment Channel (RBSC)

The connectivity proxy Helm chart is available via the RBSC Helm repository. This requires having an S-user with associated licenses.

Note

Contact your account manager to clarify access to the repository.

Pull information (for the latest version):

- **Registry:** 73554900100900005672.helmsrv.cdn.repositories.cloud.sap
- **Repository:** com.sap.cloud.connectivity/connectivity-proxy
- **Tag:** 2.12.1
- **Authorization:** see [Manage Technical Users in SAP Repositories Management](#) (RBSC documentation).

Example:

```
helm repo add connectivity https://
73554900100900005672.helmsrv.cdn.repositories.cloud.sap --username <user> --
password <pass>
helm pull connectivity/connectivity-proxy --version=2.12.1
```

[Back to Top \[page 2\]](#)

Related Information

[Operations via Helm \[page 767\]](#)

[Operations via Separate YAML Files \[page 770\]](#)

[Configuration Guide \[page 772\]](#)

[Additional Security Aspects \[page 783\]](#)

[Sizing Recommendations \[page 784\]](#)

1.3.2.1 Operations via Helm

Use the Helm chart to configure and manage the life cycle of the connectivity proxy for Kubernetes.

Note

Out of the box, the Helm chart only supports the [NGINX Ingress Controller](#) (default) and the [Istio Ingress Gateway](#) (for more information, see [Installing the Connectivity Proxy in Clusters with Istio \[page 758\]](#)).

[Deploy \[page 767\]](#)

[Update / Upgrade / Downgrade \[page 768\]](#)

[Undeploy \[page 769\]](#)

Deploy

Note

For this procedure, you must have a generated public/private TLS key pair as a prerequisite. For generating the TLS key pair, you can use [Gardener Certificate resources](#), *openssl*.

To deploy the connectivity proxy on a cluster that does not have the Helm chart yet, for example, in a new namespace, follow these steps:

1. Get the connectivity proxy Helm chart, as described in [Lifecycle Management \[page 764\]](#).
2. Create a Kubernetes secret from the generated public/private TLS key pair (for the Ingress public endpoint). Depending on the Connectivity proxy release version, the secret might require additional fields:
 1. Connectivity proxy release < 2.4.1:
Download the list of trusted CAs for the BTP region to which you are pairing the Connectivity proxy.
Example:

```
# download the list of trusted CAs
curl https://connectivity.cf.{region_domain}/api/v1/CAs -o
connectivity_ca.crt
```

```
# create a k8s secret
kubectl create secret generic connectivity-tls --from-
file=ca.crt=connectivity_ca.crt --from-file=tls.key=private.key --from-
file=tls.crt=public.crt --namespace my-namespace
```

Where `private.key` is the private key and `public.crt` is the public key of a TLS certificate, generated for the Ingress public endpoint of the connectivity proxy (the one which the Cloud Connector connects to).

→ Remember

The content of `/api/v1/CAs` may change over time. Make sure you update it regularly.

2. Connectivity proxy release `>= 2.4.1`:

Download of the trusted CAs is automated. They are saved in a secret in the respective name space. The secret's name is formed in the following way: `connectivity-ca-<helm_installation_name>`. For example, if the helm installation name is `connectivity-proxy`, the generated secret name is `connectivity-ca-connectivity-proxy`.

📌 Note

If a secret with the same name pattern, as mentioned above, is present whenever the Connectivity proxy is installed/upgraded, its content is overridden.

Example:

```
# create a k8s secret
kubectl create secret generic connectivity-tls --from-
file=tls.key=private.key --from-file=tls.crt=public.crt --namespace my-
namespace
```

Where `private.key` is the private key and `public.crt` is the public key of a TLS certificate, generated for the Ingress public endpoint of the connectivity proxy (the one which the Cloud Connector connects to).

3. Prepare the `values.yaml` file for your your scenario, as described in [Configuration Guide \[page 772\]](#).
4. Use the Helm CLI to deploy the connectivity proxy. Example:

```
helm install connectivity-proxy ./connectivity-proxy-<version>.tgz -f
values.yaml --namespace my-namespace
```

[Back to Top \[page 767\]](#)

Update / Upgrade / Downgrade

When you have a connectivity proxy deployed on the cluster, you may want to maintain it by changing configurations and/or changing its version. To do so, follow these steps:

1. Get the connectivity proxy Helm chart, as described in [Lifecycle Management \[page 764\]](#). It can be the same version as the one currently installed or a different version, when you want to upgrade or downgrade.
2. Prepare the `values.yaml` file for your your scenario, as described in [Configuration Guide \[page 772\]](#). Here you can just modify the one you used previously by applying the changes you desire.

3. Use the Helm CLI to upgrade the connectivity proxy. Example:

```
helm upgrade connectivity-proxy ./connectivity-proxy-<version>.tgz -f values.yaml --namespace my-namespace
```

Note

Each Helm upgrade will result in a restart of the connectivity proxy pod(s). This is done to ensure that configuration changes are picked up immediately.

[Back to Top \[page 767\]](#)

Undeploy

If you need to remove the connectivity proxy from your cluster or from a namespace, you can do it almost completely via normal Helm tools. However, there are some additional actions required. Please follow these steps:

1. Use the Helm CLI to undeploy the connectivity proxy. Example:

```
helm uninstall connectivity-proxy --namespace my-namespace
```

2. Delete the Kubernetes secret representing the TLS certificate for the connectivity proxy public endpoint. Example:

```
kubectl delete secret connectivity-tls --namespace my-namespace
```

Note

For Connectivity proxy releases $\geq 2.4.1$, the secret containing the trusted CAs is removed automatically when `helm uninstall` is executed.

Caution

As of connectivity proxy release 2.8, trying to execute `helm uninstall`, while there are still resources of type `ServiceMapping`, would result in an error (*job failed: BackoffLimitExceeded*). Detailed error message can be obtained after looking into the logs of the newly created pod `servicemapping-cleanup-verifier-job`, and the proxy would not be uninstalled. Also, `helm upgrade` operations would no longer be available.

To proceed, remove all resources of type `ServiceMapping` and retry the uninstallation .

[Back to Top \[page 767\]](#)

1.3.2.2 Operations via Separate YAML Files

Using separate YAML files to configure and manage the life cycle of the connectivity proxy for Kubernetes.

⚠ Caution

As operating with separate YAML files that you modify and maintain is considered error prone, we do not recommend this method. Consider using [Operations via Helm \[page 767\]](#) instead. If you do need to use separate YAML files, we recommend that you generate them via Helm template.


[Deploy \[page 770\]](#)

[Update / Upgrade / Downgrade \[page 771\]](#)

[Undeploy \[page 771\]](#)

Deploy

To deploy the connectivity proxy on a cluster that does not have it yet, for example, in a new namespace, follow these steps:

1. Get the connectivity proxy YAML files via Helm template
2. Download the list of trusted CAs for the SAP BTP region to which you want to pair the connectivity proxy. Create a Kubernetes secret from it together with a previously generated public/private TLS key pair (for the Ingress public endpoint). For generating the TLS key pair, you can use [Gardener Certificate resources](#) , `openssl`, or some other tool. Example:

```
# download the list of trusted CAs
curl https://connectivity.cf.{region_domain}/api/v1/CAs -o connectivity_ca.crt
# create a Kubernetes secret
kubectl create secret generic connectivity-tls --from-
file=ca.crt=connectivity_ca.crt --from-file=tls.key=private.key --from-
file=tls.crt=public.crt --namespace my-namespace
```

Where `private.key` is the private key and `public.crt` is the public key of a TLS certificate, generated for the Ingress public endpoint of the connectivity proxy (the one which the Cloud Connector connects to).

→ Remember

The content of `/api/v1/CAs` may change over time. Make sure you update it regularly.

3. Use the `kubectl` CLI to deploy the connectivity proxy. Example:

```
kubectl apply -f my_changed_resource.yaml --namespace my-namespace
```

[Back to Top \[page 770\]](#)

Update / Upgrade / Downgrade

When you have a connectivity proxy deployed on the cluster, you may want to maintain it by changing configurations and/or changing its version. To do so, follow these steps:

1. Get the YAML files (or only those you want to modify) you used to deploy the connectivity proxy. You can also export them from the cluster via `kubectl`.
2. Make the desired changes. For example, you can modify the subaccount ID in the `config` map or the connectivity proxy version in the deployment.
3. Use the `kubectl` CLI to apply your changes. Example:

```
kubectl apply -f my_changed_resource.yaml --namespace my-namespace
```

→ Tip

You can also use `kubectl edit` to directly modify the resources on the cluster.

ⓘ Note

When updating secrets or config maps, you must restart the pod(s) to activate the changes. You can do this by running `kubectl rollout restart statefulset/connectivity-proxy`.

[Back to Top \[page 770\]](#)

Undeploy

If you need to remove the connectivity proxy from your cluster or from a namespace, follow these steps:

1. Get the YAML files you used to deploy the connectivity proxy. You can also export them from the cluster via `kubectl`.
2. Use the `kubectl` CLI to undeploy the connectivity proxy. Example:

```
kubectl delete -f ./dir_with_yaml_files --namespace my-namespace
```

→ Tip

You can also use `kubectl delete <resource type> <resource name>`.

3. Delete the Kubernetes secret representing the TLS certificate for the connectivity proxy public endpoint. Example:

```
kubectl delete secret connectivity-tls --namespace my-namespace
```

[Back to Top \[page 770\]](#)

1.3.2.3 Configuration Guide

Find an overview of configuration parameters for the connectivity proxy for Kubernetes.

Refer to the table below for the configurations available in the `values.yaml` file of the connectivity proxy.

Note

Make sure you become familiar with the semantics, restrictions and interoperability aspects of each property before using it.

[Parameter Overview \[page 772\]](#)

[Example \[page 780\]](#)

Parameter Overview

Parameter	Description	Default
<code>chart.nameSuffix</code>	Custom string used to suffix your resources.	"" (empty string)
<code>config.displayName</code>	Display name for the connectivity proxy installation. It is displayed in the Cloud Connector UI in Cloud Connections section.	None
<code>config.integration.connectivityService.credentialsKey</code>	Key in the Connectivity service secret resource that holds the base64-encoded value of the connectivity proxy service key.	"service_key"
<code>config.servers.businessDataTunnel.enableTls</code>	Enables end-to-end mutual TLS. See Mutual TLS [page 741] for details.	false
<code>config.servers.businessDataTunnel.externalHost</code>	External ingress host of the business data tunnel server.	None
<code>config.servers.businessDataTunnel.externalPort</code>	External ingress port of the business data tunnel server.	443
<code>config.servers.businessDataTunnel.strictSniEnabled</code>	Flag for enabling and disabling strict SNI verification in the business data tunnel server.	true
<code>config.servers.businessDataTunnel.port</code>	Port on which to start the business data tunnel server.	8042

Parameter	Description	Default
config.servers.businessDataTunnel.threadPoolSize	Number of worker threads for the tunnel server. If not set, the value will be calculated, based on the available resources. The value has a lower limit of 20.	None
config.servers.proxy.authorization.oauth.allowedClientId	Oauth client ID that is authorized to pass through the connectivity proxy when authorization is enabled.	None
config.servers.proxy.clientConnection.writeTimeoutSeconds	Write timeout for the client connections to the proxy (from the proxy's point of view).	
config.servers.proxy.http.allowRemoteConnections	Flag for enabling and disabling calls from outside the pod to the HTTP proxy server.	true
config.servers.proxy.http.enabled	Flag for enabling and disabling the HTTP proxy server.	false
config.servers.proxy.http.enableProxyAuthorization	Flag for enabling and disabling the HTTP proxy authorization. See Operational Modes [page 737] for details.	true
config.servers.proxy.http.port	Port on which to start the HTTP proxy server.	20003
config.servers.proxy.rfcAndLdap.allowRemoteConnections	Flag for enabling and disabling calls from outside the pod to the RFC and LDAP proxy server.	true
config.servers.proxy.socks5.allowRemoteConnections	Flag for enabling and disabling calls from outside the pod to the SOCKS5 proxy server.	true
config.servers.proxy.socks5.enabled	Flag for enabling and disabling the SOCKS5 proxy server.	true
config.servers.proxy.socks5.enableProxyAuthorization	Flag for enabling and disabling the SOCKS5 proxy authorization. See Operational Modes [page 737] for details.	true
config.servers.proxy.socks5.port	Port on which to start the SOCKS5 proxy server.	20004
config.servers.proxy.threadPoolSize	Number of worker threads for the proxy server. If not set, the value will be calculated, based on the available resources. The value has a lower limit of 20.	None
config.jvm.errorFilePath	Directory for JVM error logs.	"/var/log/connectivity"
config.jvm.heapDumpPath	Directory for JVM heap dumps.	"/var/log/connectivity"

Parameter	Description	Default
config.jvm.memory.maxDirectSize	Maximum direct memory size to be set to the java process of the Connectivity proxy. If not set, the value will be calculated, based on the available resources. If the available resources do not set a limit, a default value is used.	512m
config.jvm.memory.maxHeapSize	Maximum heap size to be set to the java process of the Connectivity proxy. If not set, the value will be calculated, based on the available resources. If the available resources do not set a limit, a default value is used.	512m
config.jvm.memory.minHeapSize	Minimum heap size to be set to the java process of the Connectivity proxy. If not set, the value will be calculated, based on the available resources. If the available resources do not set a limit, a default value is used.	128m
config.highAvailabilityMode	High availability mode in which the connectivity proxy works. The possible values are off , subdomain , and path . See High Availability [page 749] for details.	"off"
config.subaccountId	ID of the subaccount, on whose behalf the connectivity proxy is running. This is mandatory only for the <i>dedicated</i> tenant mode. It is ignored when running in <i>shared</i> tenant mode.	None
config.subaccountSubdomain	Subdomain of the subaccount, on whose behalf the connectivity proxy is running. This is mandatory only for the <i>dedicated</i> tenant mode when there is at least one proxy with disabled authorization. It is ignored when running in <i>shared</i> tenant mode or if authorization is enabled for all proxies.	None
config.tenantMode	Mode in which the connectivity proxy works. The possible values are <i>dedicated</i> and <i>shared</i> . See Operational Modes [page 737] for details.	"dedicated"

Parameter	Description	Default
config.serviceChannels.enabled	Enables or disables the service channels which allow <i>on-premise to cloud</i> connectivity to services running in a Kubernetes cluster.	true
<div style="border-left: 2px solid orange; padding-left: 10px; margin: 10px 0;"> <p>⚠ Caution</p> <p>Only one connectivity proxy per Kubernetes cluster can have service channels enabled.</p> </div> <p>For more information, see Service Channels: On-Premise-to-Cloud Connectivity [page 756].</p>		
config.multiRegionMode.enabled	Enables or disables the multi-region mode.	false
config.multiRegionMode.configMapName	The name of the config map holding all region configurations which will be used by the connectivity proxy if multi-region mode is enabled.	None
secretConfig.integration.connectivity-Service.secretName	Name of the secret, which will hold the credentials for the Connectivity service.	"connectivity-proxy-service-key"
secretConfig.integration.connectivity-Service.secretData	base64-encoded value of the service key, obtained from the connectivity proxy service instance.	None
secretConfig.servers.businessDataTunnel.secretName	Name of the secret that is used for TLS handshake with The Ingress proxy or Cloud Connector. See Mutual TLS [page 741] for details.	None
secretConfig.servers.businessDataTunnel.secretData.caCertificate	CA certificate part of the secret that is used for TLS handshake with the Ingress proxy or Cloud Connector. See Mutual TLS [page 741] for details.	None
secretConfig.servers.businessDataTunnel.secretData.certificate	Certificate part of the secret that is used for TLS handshake with the Ingress proxy or Cloud Connector. See Mutual TLS [page 741] for details.	None
secretConfig.servers.businessDataTunnel.secretData.key	Key part of the secret that is used for TLS handshake with the Ingress proxy or Cloud Connector. See Mutual TLS [page 741] for details.	None
deployment.nodeSelector	Label(s) which are configured for the <code>nodeSelector</code> property for all connectivity proxy-associated pods. More info: https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/#nodeselector 🐾	None

Parameter	Description	Default
deployment.image.pullPolicy	One of Always , Never , IfNotPresent . For more information, see: Updating images (Kubernetes documentation).	"IfNotPresent"
deployment.image.pullSecret	Secret used for authentication against the repository.	None
deployment.image.registry	Name of the registry from which the Connectivity deployment is downloaded.	<the image repository>
deployment.image.repository	Name of the repository.	"com.sap.cloud.connectivity/connectivity-proxy"
deployment.image.tag	Version of the connectivity proxy to be deployed.	1
deployment.image.digest	Digest the connectivity proxy's image to be deployed. If both digest and tag are provided, the value of the digest has precedence.	None
deployment.utilityImage.pullPolicy	One of <i>Always</i> , <i>Never</i> , <i>IfNotPresent</i> . More info: https://kubernetes.io/docs/concepts/containers/images#updating-images	"IfNotPresent"
deployment.utilityImage.pullSecret	Pull secret for the utility image, used for authenticating against the repository.	None
deployment.utilityImage.registry	Name of the registry from which the utility image will be downloaded.	"docker.io"
deployment.utilityImage.repository	Name of the repository of the utility image.	"alpine"
<div style="border-left: 3px solid orange; padding-left: 10px;"> <p>⚠ Caution</p> <p>The repository must point to an image, which is a version of the Docker official Alpine image (see Alpine Docker Image). Usage of any other type of container image is not recommended and can result in broken or dysfunctional connectivity proxy deployment.</p> </div>		
deployment.utilityImage.tag	Version of the utility image to be deployed.	"3.18.0"
deployment.utilityImage.digest	Digest of the utility image to be deployed. If both digest and tag are provided, the value of the digest has precedence.	None

Parameter	Description	Default
deployment.replicaCount	Amount of connectivity proxies to start from this deployment.	1
deployment.resources.maxFileDescriptorCount	Maximum number of file descriptors that will be used. This value should be based on the overall scenario in which the connectivity proxy is used and on the load of the proxy (user requests to the proxy).	64000
deployment.resources.limits.cpu	The <i>kubelet</i> enforces the limit so that the running container is not allowed to use more CPU resource than set as limit. If the limit is crossed, the process is throttled.	1
deployment.resources.limits.memory	The <i>kubelet</i> enforces the limit so that the running container is not allowed to use more memory than set as limit. If the limit is crossed, the process is terminated with an out-of-memory (OOM) error.	1024M
deployment.resources.requests.cpu	The Kubernetes scheduler uses this information to decide which node to place the pod on. If there are no nodes with the specified amount of CPU resources, the pod won't be scheduled (and therefore not started).	0.1
deployment.resources.requests.memory	The Kubernetes scheduler uses this information to decide which node to place the pod on. If there are no nodes with the specified amount of memory, the pod won't be scheduled (and therefore not started).	256M
deployment.autoscaling.horizontal.enabled	Enables or disables the <i>Horizontal Pod Autoscaler</i> mechanism, see Horizontal Pod Autoscaler (Kubernetes documentation).	256M
deployment.autoscaling.horizontal.maxReplicaCount	Upper limit for the number of connectivity proxy replicas to which the autoscaler can scale up. It should be higher than <code>deployment.replicaCount</code> .	2
deployment.autoscaling.horizontal.metrics.cpuAverageUtilization	Target value of the average CPU metric across all connectivity proxy pods, represented as a percentage of the requested value of the CPU for the pods.	80
deployment.autoscaling.horizontal.metrics.memoryAverageUtilization	Target value of the average memory metric across all connectivity proxy pods, represented as a percentage of the requested value of the memory for the pods.	80

Parameter	Description	Default
deployment.autoscaling.vertical.enabled	Enables or disables the <i>Vertical Pod Autoscaler</i> mechanism.	false
	<div style="border: 1px solid orange; padding: 5px;"> <p>⚠ Caution</p> <p>To take effect, the <i>Vertical Pod Autoscaling</i> mechanism for the cluster must be enabled. See Vertical Pod Auto-Scaling for details (Github Gardener Kubernetes documentation).</p> </div>	
deployment.autoscaling.vertical.updateMode	Mode in which the <i>Vertical Pod Autoscaler</i> should operate. See Vertical Pod Autoscaler Quick start for details (Github Kubernetes documentation).	"Off"
deployment.restartWatcher.enabled	Enables or disables the restart watcher which watches for changes in secrets and configmaps containing the label <i>connectivityproxy.sap.com/restart</i> , and restarts the connectivity proxy if needed. For more information, see Automatic Pickup on Resource Changes [page 755] .	false
ingress.annotations	Annotations to add to the Ingress resource.	{}
ingress.className	Class name of the ingress resource. The possible values are "nginx" and "istio".	"nginx"
ingress.healthcheck.tls.proxy.secretName	Name of the secret that is used for TLS handshake with the connectivity proxy. See External Health Checking [page 745] for details.	None
ingress.healthcheck.tls.proxy.secretData.caCertificate	CA certificate part of the secret that is used for TLS handshake with the connectivity proxy. See External Health Checking [page 745] for details.	None
ingress.healthcheck.tls.proxy.secretData.certificate	Certificate part of the secret that is used for TLS handshake with the connectivity proxy. See External Health Checking [page 745] for details.	None
ingress.healthcheck.tls.proxy.secretData.key	Key part of the secret that is used for TLS handshake with the connectivity proxy. See External Health Checking [page 745] for details.	None
ingress.healthcheck.tls.secretName	Name of the secret that is used for TLS handshake with external health checking application. See External Health Checking [page 745] for details.	None

Parameter	Description	Default
ingress.healthcheck.tls.secretData.certificate	Certificate part of the secret that is used for TLS handshake with external health checking application. See External Health Checking [page 745] for details.	None
ingress.healthcheck.tls.secretData.key	Key part of the secret that is used for TLS handshake with external health checking application. See External Health Checking [page 745] for details.	None
ingress.istio.namespace	Installation namespace of Istio. For more information, see Installing the Connectivity Proxy in Clusters with Istio [page 758] .	"istio-system"
ingress.istio.gateway.selector	Selector that is configured in the Istio ingress gateway (used in the gateway configuration which is applied on the gateway pods). For more information, see Installing the Connectivity Proxy in Clusters with Istio [page 758] .	{ istio: ingressgateway }
ingress.istio.tls.ciphers	Allowlist of enabled TLS ciphers specified as an array of strings.	As determined by Istio: https://istio.io/latest/docs/reference/config/networking/gateway/#ServerTLSSettings , see the <code>cipherSuites</code> setting.
ingress.nginx.tls.ciphers	Enabled TLS ciphers specified in OpenSSL format.	As determined by nginx: https://nginx.org/en/docs/http/nginx_http_ssl_module.html#ssl_ciphers
ingress.timeouts.proxy.connect	Connect timeout (in seconds), for the Ingress controller.	10
ingress.timeouts.proxy.read	Read timeout (in seconds), for the Ingress controller.	120
<div style="border-left: 2px solid orange; padding-left: 10px;"> <p>⚠ Caution</p> <p>Make sure the load balancer exposing the Ingress controller is also configured to allow longer times. For example, for NGINX with AWS, you need to add a special annotation for the LB service of the Ingress controller.</p> </div>		
ingress.timeouts.proxy.send	Send timeout (in seconds), for the Ingress controller.	120

Parameter	Description	Default
ingress.tls.proxy.secretName	Name of the secret that is used for TLS handshake with the connectivity proxy. See Mutual TLS [page 741] for details.	None
ingress.tls.proxy.secretData.caCertificate	CA certificate part of the secret that is used for TLS handshake with the connectivity proxy. See Mutual TLS [page 741] for details.	None
ingress.tls.proxy.secretData.certificate	Certificate part of the secret that is used for TLS handshake with the connectivity proxy. See Mutual TLS [page 741] for details.	None
ingress.tls.proxy.secretData.key	Key part of the secret that is used for TLS handshake with the connectivity proxy. See Mutual TLS [page 741] for details.	None
ingress.tls.secretName	Name of the secret that is used for TLS handshake with the Cloud Connector.	None
ingress.tls.secretData.caCertificate	CA certificate part of the secret that is used for TLS handshake with the Cloud Connector.	None
ingress.tls.secretData.certificate	Certificate part of the secret that is used for TLS handshake with the Cloud Connector.	None
ingress.tls.secretData.key	Key part of the secret that is used for TLS handshake with the Cloud Connector.	None
transparent-proxy.enabled	Flag for enabling and disabling transparent proxy installation as a subchart.	false

[Back to Top \[page 772\]](#)

Example

⚠ Caution

The following example shows the full structure and does not represent a productive `values.yaml` file. Many properties listed here are mutually exclusive and should *not be used together* in real situations.

values.yaml structure example

```
chart:
  nameSuffix: "a string to append to resource names"
config:
```

```

integration:
  auditlog:
    mode: "service"/"console"
    serviceCredentialsKey: "specifies the filename of the mounted auditlog
secret"
  connectivityService:
    serviceCredentialsKey: "specifies the filename of the mounted connectivity
service secret"
  servers:
    businessDataTunnel:
      enableTls: true/false
      externalHost: "myconnproxy.mycluster.com"
      externalPort: 443
      strictSniEnabled: true/false
      port: 8042
    proxy:
      authorization:
        oauth:
          allowedClientId: "the client id which is allowed as JWT issuer for
proxy authorization"
      http:
        allowRemoteConnections: true/false
        enabled: true/false
        enableProxyAuthorization: true/false
        port: 20003
      rfcAndLdap:
        allowRemoteConnections: true/false
        enabled: true/false
        enableProxyAuthorization: true/false
        port: 20001
      socks5:
        allowRemoteConnections: true/false
        enabled: true/false
        enableProxyAuthorization: true/false
        port: 20004
      clientConnection:
        writeTimeoutSeconds: 10

jvm:
  errorFilePath: "directory for JVM error logs"
  heapDumpPath: "directory for JVM heap dumps"
  memory:
    maxHeapSize: 256m
    minHeapSize: 16m
  highAvailabilityMode: "off"/"subdomain"/"path"
  subaccountId: "id of the subaccount, for which the proxy is running"
  subaccountSubdomain: "subdomain of the subaccount, for which the proxy is
running"
  tenantMode: dedicated/shared
  serviceChannels:
    enabled: true/false

secretConfig:
  integration:
    auditlogService:
      secretData: "base64 encoded auditlog service key"
      secretName: "name of the secret resource, holding the auditlog secret"
    connectivityService:
      secretData: "base64 encoded connectivity_proxy service key"
      secretName: "name of the secret resource, holding the connectivity service
secret"
  servers:
    businessDataTunnel:
      secretData:
        caCertificate: "base64 encoded CA certificate"
        certificate: "base64 encoded certificate"
        key: "base64 encoded private key"
      secretName: "name of the secret"

```

```

deployment:
  image:
    pullPolicy: "IfNotPresent"
    pullSecret: "name of the pull secret fot the registry"
    registry: "the official SAP image repo"
    repository: "com.sap.cloud.connectivity/connectivity-proxy"
    tag: X.Y.Z
  replicaCount: 1
  resources:
    maxFileDescriptorCount: 64000
    limits:
      cpu: 1
      memory: 1024M
    requests:
      cpu: 0.1
      memory: 256M
  autoscaling:
    horizontal:
      enabled: true/false
      maxReplicaCount: 2
      metrics:
        cpuAverageUtilization: 80
        memoryAverageUtilization: 80
    vertical:
      enabled: true/false
      updateMode: "Off"

ingress:
  annotations:
    annotation1: value1
    annotation2: value2
  healthcheck:
  tls:
    proxy:
      secretData:
        caCertificate: "base64 encoded CA certificate"
        certificate: "base64 encoded certificate"
        key: "base64 encoded private key"
        secretName: "name of the secret"
      secretData:
        certificate: "base64 encoded certificate"
        key: "base64 encoded private key"
        secretName: "name of the secret"
  timeouts:
    proxy:
      connect: 10
      read: 120
      send: 120
  tls:
    proxy:
      secretData:
        caCertificate: "base64 encoded CA certificate"
        certificate: "base64 encoded certificate"
        key: "base64 encoded private key"
        secretName: "name of the secret"
      secretData:
        caCertificate: "base64 encoded CA certificate"
        certificate: "base64 encoded certificate"
        key: "base64 encoded private key"
        secretName: "name of the secret"
  nginx:
    tls:
      ciphers: "HIGH:!aNULL:!MD5"
  istio:
    tls:
      ciphers:
        - ECDHE-RSA-AES128-GCM-SHA256

```

```
- ECDHE-RSA-AES256-GCM-SHA384
```

```
transparent-proxy:  
  enabled: true/false
```

[Back to Top \[page 772\]](#)

1.3.2.4 Additional Security Aspects

Considerations on security for the traffic flow and configuration of the connectivity proxy for Kubernetes.

Security of Traffic Flow *within* a Cluster

This section refers to the traffic flow between a workload from a Kubernetes cluster and the connectivity proxy running in the same cluster.

The traffic between a workload running in the cluster and the HTTP and LDAP/RFC proxies is not encrypted. The traffic to the SOCKS5 proxy can be SSL-encrypted if the client application initiates an SSL connection. The HTTP and LDAP/RFC proxies are disabled by default for a connectivity proxy installation. If you want to enable them, add the following configuration in the `values.yml` file:

```
config:  
  servers:  
    proxy:  
      rfcAndLdap:  
        enabled: true  
      http:  
        enabled: true
```

Note

All business data that is transmitted between the connectivity proxy in the cluster and the Cloud Connector is sent over an SSL-encrypted tunnel.

Security of Traffic Flow *from Outside* a Cluster

This section refers to the traffic flow between the Cloud Connector and the connectivity proxy running in a Kubernetes cluster.

By default, the connections between the Cloud Connector and the Ingress load balancer in the Kubernetes cluster are SSL-encrypted. If you want to enable mutual TLS for the connections from the Ingress to the connectivity proxy or have end-to-end mutual TLS between the Cloud Connector and the connectivity proxy, follow the procedures described in [Mutual TLS \[page 741\]](#).

Security of Connectivity Proxy Configuration Data

This section refers to any security-sensitive configuration data which is required for the connectivity proxy.

The connectivity proxy installation requires some configuration data to be supplied via a Kubernetes secret.

By default, the Kubernetes secrets are stored as unencrypted base64-encoded strings and are transmitted in base64-encoded format, so they are basically accessible by persons with access to the cluster.

See the [Kubernetes documentation](#) for details on how to secure the usage of secrets in a cluster.

1.3.2.5 Sizing Recommendations

Find basic sizing guidance for the connectivity proxy for Kubernetes.

Sizing Options

The following table gives basic sizing guidance for different usage scenarios. The values listed in the **CPU** and **Memory**, and **Thread Pools** columns correspond to the properties you should define in the `values.yaml` file (for more information, see [Configuration Guide \[page 772\]](#)):

Size (S/M/L)	CPU	Memory	Thread Pools
S: The expected load is small - request concurrency and size is low	<ul style="list-style-type: none">deployment.resources.requests.cpu: 0.1deployment.resources.limits.cpu: 1	<ul style="list-style-type: none">deployment.resources.requests.memory: 256 Mdeployment.resources.limits.memory: 1024 M	<ul style="list-style-type: none">config.servers.business-DataTunnel.threadPool-Size: 20config.servers.proxy.threadPool-Size: 20
M: The expected load is medium - request concurrency and size is medium	<ul style="list-style-type: none">deployment.resources.requests.cpu: 1deployment.resources.limits.cpu: 2	<ul style="list-style-type: none">deployment.resources.requests.memory: 512 Mdeployment.resources.limits.memory: 2048 M	<ul style="list-style-type: none">config.servers.business-DataTunnel.threadPool-Size: 20config.servers.proxy.threadPool-Size: 20

Size (S/M/L)	CPU	Memory	Thread Pools
L: The expected load is large - request concurrency and size is medium or high	<ul style="list-style-type: none"> deployment.resources.requests.cpu: 2 deployment.resources.limits.cpu: 4 	<ul style="list-style-type: none"> deployment.resources.requests.memory: 1024 M deployment.resources.limits.memory: 4096 M 	<ul style="list-style-type: none"> config.servers.business-DataTunnel.threadPool-Size: 20 config.servers.proxy.threadPool-Size: 20

Relation to Operational Modes

The connectivity proxy can operate in multiple [Operational Modes \[page 737\]](#).

A major criteria is the type of tenancy used: single or multi-tenant. If multiple tenants are served, we recommend that you choose a slightly bigger size to make sure each tenant is served without precedence to any other tenant at the same time. Based on the chosen tenancy mode, the following general recommendations apply:

Tenancy Mode	Sizes to Consider
Single-Tenant	S, M
Multi-Tenant	M, L

Note

The above-mentioned sizing recommendations are related to the connectivity proxy software component, also acting as a tunnel server to which Cloud Connector instances connect. This means the Cloud Connector must be sized properly as well, see [Sizing Recommendations \[page 369\]](#) (Cloud Connector).

→ Remember

These sizing recommendation are just a direction point. There are many factors that affect the performance of the tunneling between Cloud Connector and connectivity proxy. They are closely related to the specifics of your scenario, expected regular and intermittent load, and so on.

1.3.3 Verification and Testing

Check if the connectivity proxy for Kubernetes is operational.

Once you have installed the connectivity proxy in your cluster, you can perform the following checks to verify it is running successfully.

Note

Before starting the checks, you must wait for a few seconds until all the components are started and can be consumed.

1. Execute the following command and verify that the status of the pod (pods) is **running**.

```
kubectl run verify-connectivity-proxy --image=nginx --restart=Never -it --rm -- curl -v -x "connectivity-proxy:20003" "http://virtual:1234/"
```

2. Call the external health check endpoint of the connectivity proxy to verify it is returning a successful response. You can call the endpoint in a web browser or execute the following command from the command line:

```
kubectl run verify-connectivity-proxy --image=nginx --restart=Never -it --rm -- curl -v -x "connectivity-proxy:20003" "http://virtual:1234/" -H "Proxy-Authorization: Bearer <proxy-authorization-jwt-value>"
```

Caution

If the host of the Ingress in your cluster is configured with a self-signed certificate, add the `-k` flag to `curl` to disable the SSL certificate verification.

3. Make sure you have connected a Cloud Connector to your cloud subaccount. For more information, see [Managing Subaccounts \[page 401\]](#).
 1. If you have deployed the connectivity proxy in a **single-tenant trusted mode** and have **enabled** the HTTP proxy in the `values.yml` file, execute the following command:

```
kubectl run verify-connectivity-proxy --image=nginx --restart=Never -it --rm -- curl -v -x "connectivity-proxy:20003" "http://virtual:1234/"
```

2. If you have deployed the connectivity proxy in a **single-tenant non-trusted** or **multi-tenant non-trusted mode** and have **enabled** the HTTP proxy in the `values.yml` file, execute the following command:

```
kubectl run verify-connectivity-proxy --image=nginx --restart=Never -it --rm -- curl -v -x "connectivity-proxy:20003" "http://virtual:1234/" -H "Proxy-Authorization: Bearer <proxy-authorization-jwt-value>"
```

3. If you have deployed the connectivity proxy in a **single-tenant trusted mode** and have **enabled** the HTTP proxy in the `values.yml` file, execute the following command:

```
kubectl run verify-connectivity-proxy --image=nginx --restart=Never -it --rm -- curl -v -x "connectivity-proxy:20003" "http://virtual:1234/" -H "SAP-CP-Connectivity-Service-Token: <connectivity-service-jwt-value>"
```

For more information on the process of fetching a JWT, see [Consuming the Connectivity Service \[page 214\]](#).

For more information on the different operational modes, see [Operational Modes \[page 737\]](#).

If the connectivity proxy is working as expected, you get one of these responses:

- `Access denied to system virtual:1234:` If this was a valid request, make sure you expose the system correctly in your Cloud Connector.

📌 Note

This response indicates that the business data from the cluster is successfully reaching your Cloud Connector, but the system `virtual:1234` is not exposed there.

- Response from your backend system if you have exposed it in the Cloud Connector.

📌 Note

This response confirms that the business data from the cluster is successfully reaching your backend system exposed in the Cloud Connector.

If you encounter problems with any of the above steps, please refer to [Troubleshooting \[page 792\]](#) and [Recommended Actions \[page 795\]](#) for further investigation.

1.3.4 Monitoring

Check operability, scenarios and metrics of the connectivity proxy for Kubernetes.

Basic Availability Monitoring

The basic availability check is the minimal verification you can do to make sure the connectivity proxy is alive. This check only shows if the process of the connectivity proxy is running and if it is able to handle requests. This check is also what is configured as the *liveness probe* for the Kubernetes deployment resource.

You can perform this check on-demand by invoking a simple HTTP GET request to the healthcheck endpoint of the connectivity proxy. If the response is `200 OK`, the check was *successful*. Any other response means the check *failed*. There are two ways to perform this:

- From **within the cluster**: Call `connectivity-proxy-tunnel:8042/healthcheck` and observe the result
- From **outside the cluster**: Call `https://healthcheck.<ingress host of the connectivity proxy>/healthcheck` and observe the result. See [External Health Checking \[page 745\]](#) for more details.


Scenario Monitoring

On top of the availability monitoring of the component itself, it is useful to also monitor entire scenarios. This, however, cannot be provided out of the box by the connectivity proxy as it is specific to the way you use the component. Some options you can explore for this are:

- Monitor the failure rate of requests to the connectivity proxy.
- Monitor the amount of error logs in the connectivity proxy.

- Set up a scheduled execution of a full scenario that performs end-to-end verification, including the operations done via the connectivity proxy.

Metrics

Currently, the connectivity proxy does not provide any dedicated support for metrics monitoring via [full metrics pipelines](#) .

However, you may want to use the [resource metrics pipeline](#)  to collect basic metrics and observe them, or configure alerts based on those metrics.

1.3.5 Using the Connectivity Proxy

Use the connectivity proxy for Kubernetes with different communication protocols and principal propagation (SSO).

[Overview \[page 788\]](#)

[TCP \[page 790\]](#)

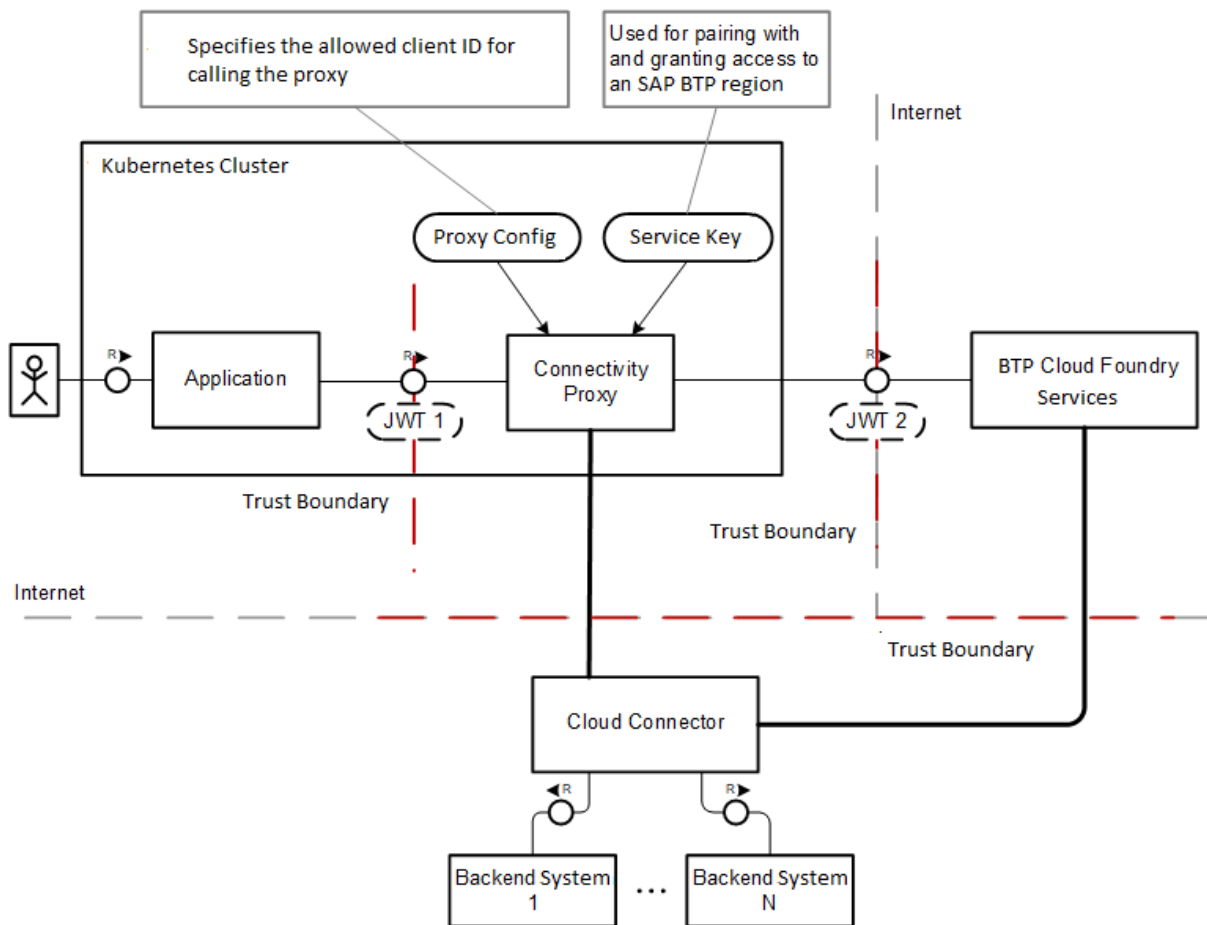
[HTTP \[page 790\]](#)

[Principal Propagation \[page 790\]](#)

[Technical User Propagation \[page 791\]](#)

Overview

The connectivity proxy offers multiple proxy endpoints which are communication protocol-specific: TCP (via SOCKS5), HTTP, RFC (invoking ABAP functions) and LDAP. Depending on the operational mode, the usage may involve technical authorization when connecting to the proxy, see [Operational Modes \[page 737\]](#).



Basically, the application needs to connect to the target on-premise system, and select the respective host and port as a protocol-standard proxy (with optional authorization) to get the request routed to the target on-premise system via the Cloud Connector.

Note

- The tunnel channel (between Cloud Connector and connectivity proxy) always uses TLS, that is, it is encrypted.
- The actual connection from the application to the connectivity proxy may be encrypted or not, depending on the exact scenario used.

Note

If a TLS connection is attempted by the application, the SOCKS5 proxy endpoint must be used.

- The actual connection from the Cloud Connector towards the on-premise system is established and controlled in the Cloud Connector. As a result, TLS can be enabled or disabled, regardless of the connection from the application to the connectivity proxy.

[Back to Top \[page 2\]](#)

TCP

TCP connectivity is achieved via the [SOCKS5](#) proxy protocol. As the authorization is based on OAuth, we provide a custom authorization scheme for SOCKS5 that lets the application pass the required OAuth tokens and establish technical authorization with the connectivity proxy. For more information, see [Using the TCP Protocol for Cloud Applications \[page 234\]](#).

[Back to Top \[page 2\]](#)

HTTP

Uses a standard HTTP proxy, just like using a corporate proxy to reach out to the Internet. For more information, see [Authentication to the On-Premise System \[page 223\]](#).

```
curl --proxy "connectivity-proxy-host:20003" /  
http://my-virtual-host:1234/my-path#my-fragment?my-query=my-value /  
--header "Proxy-Authorization: Bearer <token>"
```

[Back to Top \[page 2\]](#)

Principal Propagation

Principal propagation, also known as *user propagation*, lets you perform single sign-on (SSO) authentication of the cloud user towards an on-premise system.

The cloud user identity is passed as a token represented by a JSON Web token (JWT). It is forwarded via the connectivity proxy to the Cloud Connector, which validates and further processes it to establish SSO with the on-premise system.

For more information, see [Authenticating Users against On-Premise Systems \[page 419\]](#) and [Set Up Trust \[page 420\]](#).

As of connectivity proxy release 2.1.1, support for principal propagation with IAS tokens is added.

Prerequisites:

- Cloud Connector 2.13 (or higher) must be used.
- The Cloud Connector must be connected to a subaccount, that is, configured with the IAS tenant issuing the tokens (for more information, see [Establish Trust and Federation Between SAP Authorization and Trust Management Service and Identity Authentication](#)).

Principal propagation is supported for both HTTP and RFC communication protocols and can be used in all [Operational Modes \[page 737\]](#) of the connectivity proxy:

- **Single-tenant in trusted environment**
The user should be propagated via `SAP-Connectivity-Authentication` header.
- **Multi-tenant in trusted environment**
The user should be propagated only via `SAP-Connectivity-Authentication` header.
The `SAP-CP-Connectivity-Service-Token` header *must* be sent for handling the tenant context.
- **Single or multi-tenant in non-trusted environment**
The user should be propagated via either `Proxy-Authorization` or `SAP-Connectivity-Authentication` header, exclusively. If `Proxy-Authorization` is used, it must hold a JWT that results from a token exchange (see [JWT Bearer Grant Type](#)), to include both the context for technical access toward the Connectivity Proxy as well as the user identity.

Note

If IAS tokens are used, they can *only* be provided via the `SAP-Connectivity-Authentication` header.

Note

If in a **non-trusted environment** the user context is provided via the `Proxy-Authorization` header, the `SAP-Connectivity-Authentication` *must not* be sent.

Caution

In principal propagation scenarios, you must omit the `Authorization` header.

For more information, see also [XSUAA Token Client and Token Flow API](#).

[Back to Top \[page 2\]](#)

Technical User Propagation

Technical user propagation lets you perform single sign-on (SSO) authentication of a cloud technical user towards an on-premise system. It is very similar to *principal propagation*, but instead of forwarding the identity of business users, the identity of technical users is forwarded.

The technical user is represented by a token in form of a JSON web token (JWT) that is usually obtained via the `client_credentials` OAuth flow. It is forwarded via the connectivity proxy to the Cloud Connector, which validates and processes it to perform SSO with the on-premise system.

Technical user propagation is supported as of connectivity proxy version 2.6.1. As a prerequisite, Cloud Connector version 2.15 or higher must be used.

Technical user propagation is supported for both HTTP and RFC communication protocols and can be used in all [Operational Modes \[page 737\]](#).

The token representing the technical user must be sent to the connectivity proxy via the `SAP-Connectivity-Technical-Authentication` HTTP request header.

⚠ Caution

In *technical user propagation* scenarios, you must omit the `Authorization` header.

For more information, see [Configuring Principal Propagation \[page 420\]](#) and [XSUAA Token Client and Token Flow API ↗](#).

[Back to Top \[page 2\]](#)

1.3.6 Troubleshooting

Find procedures to troubleshoot issues with the connectivity proxy for Kubernetes.

[Get Logs of the Connectivity Proxy \[page 792\]](#)

[Changing Log Level\(s\) of the Connectivity Proxy \[page 792\]](#)

[Viewing Logger\(s\) of the Connectivity Proxy \[page 793\]](#)

[Troubleshooting the Cloud Connector \[page 794\]](#)

[Common Issues and Solutions \[page 794\]](#)

Get Logs of the Connectivity Proxy

As the connectivity proxy workload is represented as a [standard Kubernetes StatefulSet ↗](#), fetching the logs is done in the standard Kubernetes way. Example via `kubectl`:

```
kubectl logs statefulset/connectivity-proxy
```

[Back to Top \[page 2\]](#)

Changing Log Level(s) of the Connectivity Proxy

When the default logging level is not sufficient for debugging the issue you are facing, you can change the log level to get more insight about the problem.

Note

Changing a log level to something more verbose will have a negative impact on the performance of the connectivity proxy. Thus, we recommend that you do not keep such a log level for a long period of time.

Changing a log level is done without any downtime and requires no restarts. All you need to do is invoke a simple command on a pod of the connectivity proxy. Here are some examples:

Put all loggers on DEBUG (full command)

```
kubectl exec <pod> -it -- connectivity-proxy-operations logging change-log-level DEBUG
```

Put all loggers on DEBUG (shortcut)

```
kubectl exec <pod> -it -- change-log-level DEBUG
```

Put only some loggers on DEBUG (shortcut)

```
kubectl exec <pod> -it -- change-log-level com.sap.core.connectivity.tunnel.k8s  
DEBUG # Will affect all loggers with names, starting with  
com.sap.core.connectivity.tunnel.k8s
```

Restore log level to default (shortcut)

```
kubectl exec <pod> -it -- change-log-level INFO
```

For more information, check the help text of the command:

```
kubectl exec <pod> -it -- change-log-level help
```

[Back to Top \[page 2\]](#)

Viewing Logger(s) of the Connectivity Proxy

List all loggers (full command)

```
kubectl exec <pod> -it -- connectivity-proxy-operations logging list-loggers
```

List all loggers (shortcut)

```
kubectl exec <pod> -it -- list-loggers  
# or
```



```
kubectl exec <pod> -it -- get-loggers
```

List only some specific loggers (shortcut)

```
kubectl exec <pod> -it -- list-loggers com.sap.core.connectivity.tunnel.k8s  
# Will show all loggers with names, starting with  
com.sap.core.connectivity.tunnel.k8s
```

For more information, check the help text of the command:

```
kubectl exec <pod> -it -- list-loggers help
```

[Back to Top \[page 2\]](#)

Troubleshooting the Cloud Connector

For troubleshooting the Cloud Connector, see [Troubleshooting \[page 694\]](#) (Cloud Connector).

[Back to Top \[page 2\]](#)

Common Issues and Solutions

Issue	Solution
You use the HTTP proxy (port 20003) and get a 405 response.	Make sure the URL you are calling is <code>http://<virtual host>:<virtual port></code> and not <code>https://<virtual host>:<virtual port></code> .
You use the HTTP proxy (port 20003) and get a 407 response (in a non-trusted environment / proxy authorization turned on).	Make sure you set the <code>Proxy-Authorization</code> header with a value in the format <code>"Bearer <valid JWT token>"</code> .
You use the HTTP proxy (port 20003) and get a 503 response stating that there is no Cloud Connector for your subaccount.	<ul style="list-style-type: none">• Make sure the Cloud Connector you are targeting is still connected.• Make sure it's location ID matches the one used in the request.• Make sure the Cloud Connector is connected to the same subaccount, on whose behalf the token for the request is issued.

Issue	Solution
Out of nowhere, the connectivity proxy stops working / monitors indicate a failure (considered an outage if it happens in production).	See Recommended Actions [page 795] .
An SSL error is shown in the Cloud Connector when trying to send a request through the connectivity proxy.	<p>Assuming the SSL error occurs during the call from the Cloud Connector to the public endpoint of the connectivity proxy, there are two options:</p> <ul style="list-style-type: none"> • Error is due to a connection termination during the SSL handshake. In this case, check you intermediate components (proxies, firewalls, etc.). • The JVM on which the Cloud Connector is running does not trust the TLS certificate of the public endpoint of the connectivity proxy. In this case, import the certificate (or its CA) into the trust store of the JVM.

[Back to Top \[page 2\]](#)

Related Information

[Recommended Actions \[page 795\]](#)

1.3.6.1 Recommended Actions

Find procedures to resolve an outage of the connectivity proxy for Kubernetes functionality.

Caution

Before performing any of the steps below, make sure there is really an outage of the connectivity proxy and not just a general problem on the entire cluster.

[Pre-Intervention Steps \[page 796\]](#)

[Recovery Attempt \[page 796\]](#)

[Request Help from SAP \[page 798\]](#)

[Request Root Cause Analysis \(RCA\) \[page 798\]](#)

Pre-Intervention Steps

Before doing any restarts or modifications, it is important that you collect all relevant information.

1. Check the status of the connectivity proxy pods and collect the outcome. Example via `kubectl`:

```
kubectl get pods | grep 'connectivity-proxy'
```

2. Perform basic availability monitoring from *within* the cluster, as described in [Monitoring \[page 787\]](#), and collect the outcome. This can be done from an existing container or by spinning up a container for the check. Example:

```
kubectl run perform-hc --image=curlimages/curl -it --rm --restart=Never --curl -vvv 'connectivity-proxy-tunnel:8042/healthcheck'
```

3. Perform basic availability monitoring from *outside* the cluster, as described in [Monitoring \[page 787\]](#), and collect the outcome. This can be done from your browser or via REST clients like cURL or Postman. Example:

```
curl -vvv 'https://healthcheck.connectivitytunnel.ingress.mycluster.com/healthcheck'
```

4. Collect the logs of the connectivity proxy (see [Troubleshooting \[page 792\]](#)).
5. Proceed to **Recovery Attempt**.

[Back to Top \[page 795\]](#)

Recovery Attempt

The exact action to take here depends on the check result in steps 2 and 3 of section [Pre-Intervention Steps \[page 796\]](#). Choose one of the four options below, according to the outcome of your checks.

→ Tip

Check if the cause of the outage might be insufficient resources. For more information, see [Sizing Recommendations \[page 784\]](#). If this is the possible cause, try scaling the connectivity proxy vertically and/or horizontally (see [Configuration Guide \[page 772\]](#)).

- Option 1: [Check Succeeds from within the Cluster and Fails from outside the Cluster \[page 796\]](#)
- Option 2: [Check Fails from within the Cluster and Succeeds from outside the Cluster \[page 797\]](#)
- Option 3: [Check Fails from within the Cluster and Fails from outside the Cluster \[page 797\]](#)
- Option 4: [Check Succeeds from within the Cluster and Succeeds from outside the Cluster \[page 797\]](#)

Check Succeeds from within the Cluster and Fails from outside the Cluster

This indicates some sort of issue with the Ingress configuration. Some possible reasons:

- TLS certificate has expired.
- Something went wrong with the Ingress controller.

Such a situation is likely not an issue with the connectivity proxy. Next steps should be to stop following the steps here and shift focus towards the Ingress configuration and Ingress controller.

[Back to Recovery Attempt \[page 796\]](#)

[Back to Top \[page 795\]](#)

Check *Fails* from within the Cluster and *Succeeds* from outside the Cluster

This indicates some sort of issue with the exposure of the connectivity proxy to internal pods. Some possible reasons:

- Some unwanted network policy came into effect, preventing calls to the connectivity proxy from where you are executing them.

Such a situation is likely not an issue with the connectivity proxy. Next steps should be to stop following the steps here and shift focus towards cluster configurations and the network policies that affect access to the connectivity proxy.

[Back to Recovery Attempt \[page 796\]](#)

[Back to Top \[page 795\]](#)

Check *Fails* from within the Cluster and *Fails* from outside the Cluster

This indicates that the connectivity proxy itself is indeed having issues. Please perform the following steps:

1. Restart the connectivity proxy deployment. Example via `kubectl`:

```
kubectl rollout restart statefulset/connectivity-proxy
```

2. Collect logs from the connectivity proxy after the restart completes.
3. Check if outage is still ongoing:
 1. If no, issue is resolved, proceed with [Request Root Cause Analysis \(RCA\) \[page 798\]](#).
 2. If yes, issue is not resolved, proceed with [Request Help from SAP \[page 798\]](#).

[Back to Recovery Attempt \[page 796\]](#)

[Back to Top \[page 795\]](#)

Check *Succeeds* from within the Cluster and *Succeeds* from outside the Cluster

This indicates that the connectivity proxy is currently considered operational, however it might still have trouble when used for real scenarios (depends on how you detect the outage).

1. Check if the outage is still ongoing:
 - If no, issue is resolved, proceed with [Request Root Cause Analysis \(RCA\) \[page 798\]](#).
 - If yes, issue is not resolved, proceed with the next step.

- Restart the connectivity proxy deployment. Example via `kubectl`:

```
kubectl rollout restart statefulset/connectivity-proxy
```

- Collect logs from the connectivity proxy after the restart completes.
- Check if outage is still ongoing:
 - If no, issue is resolved, proceed with [Request Root Cause Analysis \(RCA\) \[page 798\]](#).
 - If yes, issue is not resolved, proceed with [Request Help from SAP \[page 798\]](#).

[Back to Recovery Attempt \[page 796\]](#)

[Back to Top \[page 795\]](#)

Request Help from SAP

If you cannot resolve the issue and require help from SAP, follow this procedure:

- Open an incident on the support component (see [Connectivity Support \[page 869\]](#)) for the connectivity proxy (with the appropriate priority and impact stated).
- Provide all the collected information in the incident, including all the logs, timestamps of the events that occurred, summary of the taken actions, version of the connectivity proxy you are using, and so on.
- Engage your SAP contacts to help with this.
- Continue working in parallel to identify as much information as possible or to fine a temporary measure to mitigate the outage.

[Back to Top \[page 795\]](#)

Request Root Cause Analysis (RCA)

Once the issue is resolved, the next step is figuring out what exactly caused the issue and if there is something that can be done to prevent it from happening in the future. Follow this procedure for requesting RCA for the issue you experienced:

- Open an incident on the support component for the connectivity proxy (see [Connectivity Support \[page 869\]](#)).
- Provide all the collected information in the incident, including all the logs, timestamps of the events that occurred, summary of the taken actions, version of the connectivity proxy you are using, and so on.
- The incident will be handled according to the SAP incident SLAs.

[Back to Top \[page 795\]](#)

1.3.7 Frequently Asked Questions

Answers to the most common questions about the connectivity proxy for Kubernetes.

When using one of the two untrusted operational modes, what is the purpose of the `allowedClientId` property? What token should I provide in the `Proxy-Authorization` header?

The `Proxy-Authorization` header serves as a way for workloads calling the proxy to authenticate against it. To do this, they need to provide this header with a JSON Web token (JWT) as value.

To accept the JWT, it must be issued by the OAuth client, specified via the `allowedClientId`. By configuring the `allowedClientId`, you determine which OAuth client protects the proxy endpoints.

The OAuth client must be XSUAA-based.

How is the `Proxy-Authorization` header verified?

There are several aspects that are being verified:

- The signature is being verified by calling XSUAA to get the public keys for the tenant, on behalf of which the JWT is issued, and using those keys to check if it is valid.
- The token validity is also verified and expired tokens are rejected.
- Also, we allow only tokens issued by a specified client ID (via the `allowedClientId` configuration).

Why do I need the `connectivity:connectivity_proxy` service key?

The connectivity proxy is a distributed software component that needs to connect to an instance of the central Connectivity service to function.

This pairing is achieved via a `connectivity:connectivity_proxy` service key, which contains both routing information and credentials for this pairing.

Why do I need a public endpoint for the connectivity proxy?

To preserve the integrity of an on-premise landscape and not expose anything from there to the Internet, the flow for establishing the connection between connectivity proxy and Cloud Connector is initiated by the Cloud Connector.

The public endpoint is used by the Cloud Connector to call the connectivity proxy and enable the data exchange.

What is the relation of the connectivity proxy to the Connectivity service in SAP BTP?

The connectivity proxy is a distributed component that must be paired to an instance of the Connectivity service in SAP BTP in order to function.

Cloud Connectors would still connect to the Connectivity service on SAP BTP and the connectivity proxy will make use of those Cloud Connectors via the established pairing.

What is the relation of the connectivity proxy to the Destination service in SAP BTP?

There is no dependency or tight integration.

You can use the Destination service to store and retrieve on-premise destination configurations which can then be used to construct a request to on-premise systems through the connectivity proxy.

Are there any client libraries that I can use with the connectivity proxy?

Please check [Using the Connectivity Proxy \[page 788\]](#).

When do I need a subscription to the connectivity:connectivity_proxy instance?

Please check [Connectivity Service \[page 753\]](#).

Can I port a cloud SDK application from the Cloud Foundry environment to Kubernetes and use the connectivity proxy?

It is a bit clunky, but yes.

If you set up the connectivity proxy in an untrusted operational mode, the way the SDK works is well suited for it.

However, since the SDK is created with the Cloud Foundry environment in mind, you would need to simulate the `VCAP_SERVICES` environment to get it working.

Is there an equivalent to the lite plan from the Cloud Foundry environment? Is the lite plan relevant for the connectivity proxy?

The lite plan is only relevant for the Cloud Foundry environment.

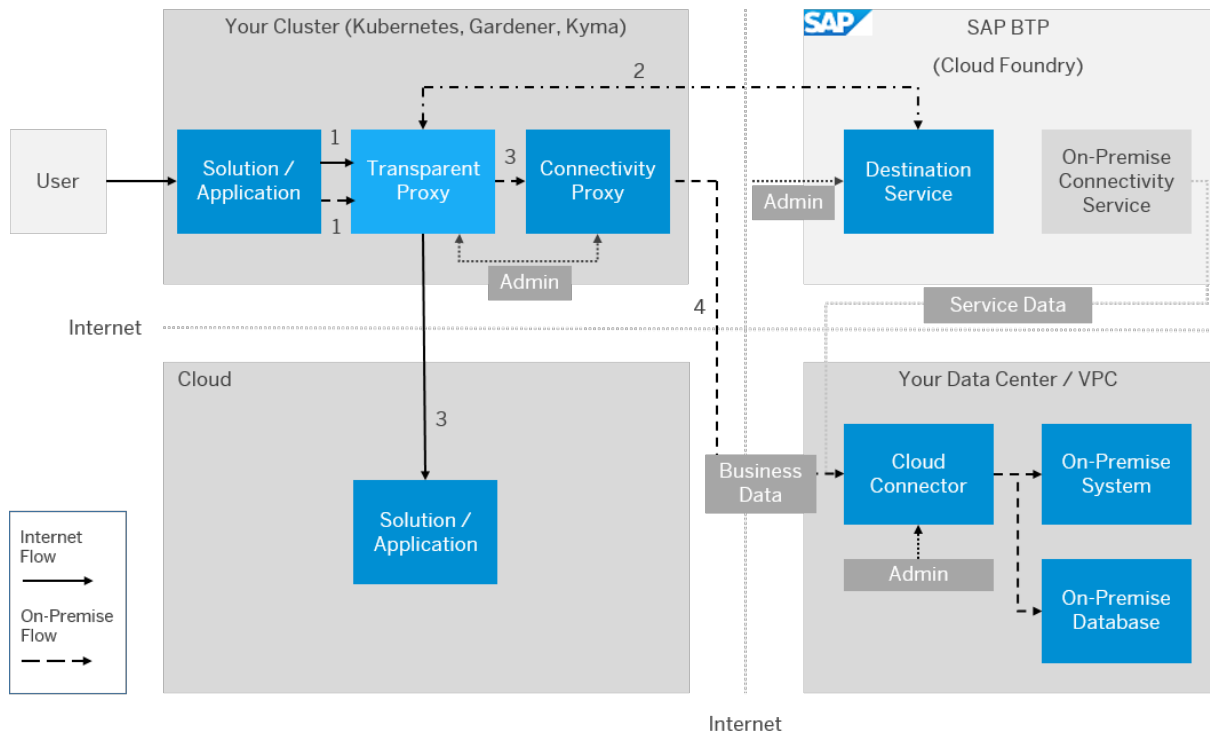
There is no lite plan for the connectivity proxy. Instead, you use an XSUAA-based OAuth client of your choice to protect the connectivity proxy, and use it to issue tokens.

1.4 Transparent Proxy for Kubernetes

Use the transparent proxy for Kubernetes to connect workloads on a Kubernetes cluster to Internet and on-premise applications.

The transparent proxy routes to SAP BTP destinations configured in the Destination service. On-premise applications must be exposed via [Cloud Connector \[page 343\]](#) (installed in the same network right next to the on-premise system) and [Connectivity Proxy for Kubernetes \[page 732\]](#) (installed in the Kubernetes cluster).

The transparent proxy is delivered as Docker images and a Helm chart. You need to run the image on your Kubernetes cluster with appropriate configurations. The Helm chart simplifies the installation process.



The transparent proxy handles HTTP(s) protocol for Internet and HTTP/TCP protocols for on-premise destinations by creating a Kubernetes service for each destination configuration. When an application tries to reach a desired system defined as a destination configuration, the transparent proxy intercepts the traffic (1), calls the Destination service to obtain the configuration for the requested destination (2), enriches the request by providing the SOCKS5 handshake for TCP and the authentication mechanism for HTTP, and routes the traffic to the desired remote system:

- Directly for Internet-accessible solutions/applications/systems (3)
- Via the connectivity proxy (3) and the Cloud Connector (4) for on-premise systems

Related Information

- [Using the Transparent Proxy \[page 802\]](#)
- [Multitenancy \[page 815\]](#)
- [Destination Custom Resource Scope \[page 816\]](#)
- [Integration with SAP BTP Connectivity \[page 817\]](#)
- [Lifecycle Management \[page 819\]](#)
- [Monitoring \[page 840\]](#)
- [Troubleshooting \[page 841\]](#)
- [Recommended Actions \[page 854\]](#)
- [Resilience \[page 857\]](#)
- [Verification and Testing \[page 858\]](#)
- [Transparent Proxy in the Kyma Environment \[page 860\]](#)

1.4.1 Using the Transparent Proxy

Use the transparent proxy for Kubernetes in different SAP BTP communication scenarios.

The transparent proxy lightens the way your Kubernetes workloads use SAP BTP destinations of type *Internet* and *on-premise*. It provides authentication, principal propagation, SOCKS5 handshake, and easy access to the destination target systems by exposing them as [Kubernetes services](#) .

To target a given destination for handling by the transparent proxy, you should create a custom Kubernetes resource of type `destinations.destination.connectivity.api.sap` that references the given destination. If you do not create a custom Kubernetes resource of type `destinations.destination.connectivity.api.sap` for a given destination, the destination will not be handled by the transparent proxy.

Note

There should be no Kubernetes service in the namespace where the transparent proxy is installed with a `name` equal to the `name` of the destination custom resource of type `destination.connectivity.api.sap`, because the transparent proxy handles the creation of Kubernetes services based on the information stored in these custom resources.

Related Information

[Internet Connectivity \[page 802\]](#)

[On-Premise Connectivity \[page 807\]](#)

[HTTP Authentication \[page 813\]](#)

[Managed Namespaces Mode \[page 815\]](#)

1.4.1.1 Internet Connectivity

Use the transparent proxy for Kubernetes to set up connections of type *Internet*.

The transparent proxy handles the HTTP(s) communication protocol for Internet destinations. As an application developer, you must create an SAP BTP destination of type `HTTP` and proxy type `Internet`, for example:

Sample Code

```
{
```

```

    "Name": "example-dest-client-cert",
    "Type": "HTTP",
    "ProxyType": "Internet",
    "URL": "https://myapp.com",
    "Authentication": "ClientCertificateAuthentication",
    "KeyStorePassword": "Abcd1234",
    "KeyStoreLocation": "cert.jks"
  }

```

To target the destination with the name "example-dest-client-cert" for handling by the transparent proxy, you should create the following Kubernetes resource in a namespace of your choice:

```

apiVersion: destination.connectivity.api.sap/v1
kind: Destination
metadata:
  name: example-dest
spec:
  destinationRef:
    name: "example-dest-client-cert"
  destinationServiceInstanceName: dest-service-instance-example // can be
omitted if config.destinationService.defaultInstanceName is provided

```

The transparent proxy monitors the available destinations in SAP BTP and compares them with the existing `destination.connectivity.api.sap` Kubernetes resources in the namespace where it is installed. Once a new SAP BTP destination is created for which a `destination.connectivity.api.sap` resource exists, the transparent proxy changes its configuration.

After the transparent proxy has executed successfully all necessary operations on the given SAP BTP destination, the status of the `destination.connectivity.api.sap` Kubernetes resource will be updated as shown below, and the DNS record "example-dest" will route to the configured URL in the destination:

```

apiVersion: destination.connectivity.api.sap/v1
kind: Destination
metadata:
  name: example-dest
spec:
  destinationRef:
    name: "example-dest-client-cert"
  destinationServiceInstanceName: dest-service-instance-example // can be
omitted if config.destinationService.defaultInstanceName is provided
status:
  conditions:
  - lastUpdateTime: "2022-09-28T07:26:46Z"
    message: Transparent Proxy is configured and Kubernetes service with name
"example-dest" is created.
    reason: ConfigurationSuccessful
    status: "True"
    type: Available

```

Once done, the application can start consuming the destination from within the Kubernetes cluster, for example:

Sample Code

```
curl example-dest.<destination-cr-namespace>
```

Note

The namespace is optional if you have created the destination custom resource (CR) in the namespace of the application that will request it. For more information, see [Kubernetes Namespaces and DNS](#).

HTTP

Mandatory destination configuration fields:

Property	Description
Name	Destination name. Must be unique for the destination level and must contain only letters, numbers, or hyphen "-".
Type	Destination type. Use HTTP for all HTTP(S) destinations.
URL	URL of the target endpoint.
Authentication	One of the listed authentication types in HTTP Destinations [page 87]
ProxyType	Internet

Additional destination configuration fields:

Property	Description
URL.connectionTimeoutInSeconds	<p>Period of time in which the HTTP client expects a confirmation after it initiated a new TCP connection.</p> <p>It must be in the range of 1 second to 5 minutes(300 seconds).</p>

Note

If you do not use it, the default value is 30 seconds. If you decide to use it, the total timeout of the initial request may be a bit longer than your chosen value, depending on the latency of calls to the Destination Service, which are needed to proxy the request.

Property	Description
URL.socketReadTimeoutInSeconds	<p data-bbox="804 349 1355 405">Period of time the HTTP client will wait for receiving a response (data) from the server.</p> <p data-bbox="804 427 1355 483">It must be in the range of 1 second to 12 hours (43200 seconds).</p> <div data-bbox="804 506 1396 763"><p data-bbox="826 521 927 551">Note</p><p data-bbox="826 577 1326 600">If you do not use it, the default value is 30 seconds.</p><p data-bbox="826 611 1370 741">If you decide to use it, the total timeout of the initial request may be a bit longer than your chosen value, depending on the latency of calls to the Destination Service, which are needed to proxy the request.</p></div>

Property**Description**

URL.headers.<header-key>

A static key prefix is used as a namespace grouping of the URL's HTTP headers. Its values are sent to the target endpoint. For each HTTP header's key, you must add a 'URL.headers' prefix separated by dot-delimiter.

For example:

Sample Code

```
{
  ...
  "URL.headers.<header-key-1>" :
  "<header-value-1>",
  ...
  "URL.headers.<header-key-N>" :
  "<header-value-N>",
}
```

Caution

If there are duplicate header keys passed from the request and contained in the destination, they are both added to the request against the target system.

Sample Code

```
{
  // other destination properties
  "URL.headers.foo": "bar"
}
```

Sample Code

```
curl -H "foo: bar2"
targetsystem -vvv
* Trying <some ip>...
* Connected to targetsystem
(<some ip>) port 80 (#0)
> GET / HTTP/1.1
> Host: targetsystem
> User-Agent: curl/7.84.0
> foo: bar2
> foo: bar
```

Property	Description
URL.queries.<query-key>	A static key prefix is used as namespace grouping of the URL's query parameters. Their values are sent to the target endpoint. For each query parameter's key you must add to a 'URL.queries' prefix, separated by dot-delimiter.

For example:

Sample Code

```
{
  ...
  "URL.queries.<query-key-1>" :
  "<query-value-1>",
  ...
  "URL.queries.<query-key-N>" :
  "<query-value-N>",
}
```

Caution

If there are duplicate header keys passed from the request and contained in the destination, they are both added to the request against the target system.

Sample Code

```
{
  /// other destination properties
  "URL.queries.foo": "bar"
}
```

Sample Code

```
curl targetsystem?foo=bar2 //
will result to the following
final URL : targetsystem?
foo=bar2&foo=bar
```

1.4.1.2 On-Premise Connectivity

Use the transparent proxy for Kubernetes to set up connections of type `on-premise`.

The transparent proxy handles both HTTP and TCP communication protocols for on-premise destinations. To use the on-premise scenarios, as an application developer you need to install a connectivity proxy and integrate

it with the transparent proxy. This happens in the transparent proxy configuration integration section. For example:

Sample Code

```
integration:
  connectivityProxy:
    serviceName: connectivity-proxy.<namespace>
    serviceCredentials:
      secretName: <the secret that contains the connectivity proxy instance
credentials>
      secretNamespace: <the secret namespace>
```

Then, you must create an SAP BTP destination of proxy type `OnPremise`, for example:

Sample Code

```
{
  "Name": "example-dest-onprem",
  "Type": "HTTP",
  "ProxyType": "OnPremise",
  "URL": "http://virtualhost:4321",
  "Authentication": "PrincipalPropagation"
}
```

To target the destination with name "example-dest-onprem" for handling by the transparent proxy, you should create the following Kubernetes resource in a namespace of your choice:

```
apiVersion: destination.connectivity.api.sap/v1
kind: Destination
metadata:
  name: example-dest
spec:
  destinationRef:
    name: "example-dest-onprem"
  destinationServiceInstanceName: dest-service-instance-example // can be
omitted if config.destinationService.defaultInstanceName is provided
```

After the transparent proxy executes successfully all necessary operations on the given SAP BTP destination, the status of the `destination.connectivity.api.sap` Kubernetes resource will be updated as shown below, and the DNS record "example-dest" will route to the configured URL in the destination:

```
apiVersion: destination.connectivity.api.sap/v1
kind: Destination
metadata:
  name: example-dest
spec:
  destinationRef:
    name: "example-dest-onprem"
  destinationServiceInstanceName: dest-service-instance-example // can be
omitted if config.destinationService.defaultInstanceName is provided
status:
  conditions:
  - lastUpdateTime: "2022-09-28T07:26:46Z"
```

```
message: Transparent Proxy is configured and Kubernetes service with name
"example-dest" is created.
reason: ConfigurationSuccessful
status: "True"
type: Available
```

Once done, the application can start consuming the destination from within the Kubernetes cluster, for example:

Sample Code

```
curl example-dest.<destination-cr-namespace>
```

Note

The namespace is optional if you have created the destination custom resource (CR) in the namespace of the application that will request it. For more information, see [Kubernetes Namespaces and DNS](#).

HTTP

Mandatory destination configuration fields:

Property	Description
Name	Destination name. Must be unique for the destination level and must contain only letters, numbers, or hyphen "-".
Type	Destination type. Use HTTP for all HTTP(S) destinations.
URL	Virtual URL of the protected on-premise application.
Authentication	One of the listed authentication types in HTTP Destinations [page 87]
ProxyType	Internet or OnPremise

Additional destination configuration fields:

Property	Description
URL.connectionTimeoutInSeconds	<p data-bbox="804 338 1382 394">Period of time in which the HTTP client expects a confirmation after it initiated a new TCP connection.</p> <p data-bbox="804 416 1382 472">It must be in the range of 1 second to 5 minutes(300 seconds).</p> <div data-bbox="804 495 1398 757" style="background-color: #f0f0f0; padding: 10px;"> <p data-bbox="826 510 927 539">Note</p> <p data-bbox="826 562 1382 734">If you do not use it, the default value is 30 seconds. If you decide to use it, the total timeout of the initial request may be a bit longer than your chosen value, depending on the latency of calls to the Destination Service, which are needed to proxy the request.</p> </div>
URL.socketReadTimeoutInSeconds	<p data-bbox="804 786 1382 842">Period of time the HTTP client will wait for receiving a response (data) from the server.</p> <p data-bbox="804 864 1382 920">It must be in the range of 1 second to 12 hours (43200 seconds).</p> <div data-bbox="804 943 1398 1205" style="background-color: #f0f0f0; padding: 10px;"> <p data-bbox="826 958 927 987">Note</p> <p data-bbox="826 1010 1382 1182">If you do not use it, the default value is 30 seconds. If you decide to use it, the total timeout of the initial request may be a bit longer than your chosen value, depending on the latency of calls to the Destination Service, which are needed to proxy the request.</p> </div>

Property**Description**

URL.headers.<header-key>

A static key prefix is used as a namespace grouping of the URL's HTTP headers. Its values are sent to the target endpoint. For each HTTP header's key, you must add a 'URL.headers' prefix separated by dot-delimiter.

For example:

Sample Code

```
{
  ...
  "URL.headers.<header-key-1>" :
  "<header-value-1>",
  ...
  "URL.headers.<header-key-N>" :
  "<header-value-N>",
}
```

Caution

If there are duplicate header keys passed from the request and contained in the destination, they are both added to the request against the target system.

Sample Code

```
{
  // other destination properties
  "URL.headers.foo": "bar"
}
```

Sample Code

```
curl -H "foo: bar2"
targetsystem -vvv
* Trying <some ip>...
* Connected to targetsystem
(<some ip>) port 80 (#0)
> GET / HTTP/1.1
> Host: targetsystem
> User-Agent: curl/7.84.0
> foo: bar2
> foo: bar
```

Property	Description
URL.queries.<query-key>	A static key prefix is used as namespace grouping of the URL's query parameters. Their values are sent to the target endpoint. For each query parameter's key you must add to a 'URL.queries' prefix, separated by dot-delimiter.

For example:

↔ Sample Code

```
{
  ...
  "URL.queries.<query-key-1>" :
  "<query-value-1>",
  ...
  "URL.queries.<query-key-N>":
  "<query-value-N>",
}
```

⚠ Caution

If there are duplicate header keys passed from the request and contained in the destination, they are both added to the request against the target system.

↔ Sample Code

```
{
  /// other destination properties
  "URL.queries.foo": "bar"
}
```

↔ Sample Code

```
curl targetsystem?foo=bar2 //
will result to the following
final URL : targetsystem?
foo=bar2&foo=bar
```

TCP

Mandatory destination configuration fields:

Property	Description
Name	Destination name.
Type	Destination type. Use TCP.
Address	<destination_host>:<destination_port> of the protected on-premise application.
ProxyType	OnPremise

SOCKS5

The connectivity proxy provides a SOCKS5 proxy that you can use to access on-premise systems via TCP-based protocols. SOCKS5 is the industry standard for proxying TCP-based traffic. For more information, see [RFC 1928](#) .

The transparent proxy performs the SOCKS5 handshake with the [Connectivity Proxy \[page 732\]](#) to enable the out-of-the-box consumption of that (otherwise complex) functionality for the application developer.

For more information on how to directly set up the usage of the TCP protocol for cloud applications, see [Using the TCP Protocol for Cloud Applications \[page 234\]](#).

1.4.1.4 HTTP Authentication

HTTP authentication for the transparent proxy for Kubernetes.

The transparent proxy enriches your request by adding the credentials configured in the destination configuration. Depending on the configured authentication type, the transparent proxy propagates them via an `Authorization` or `SAP-Connectivity-Authentication` header. For more information, see [Configuring Authentication \[page 87\]](#).

HTTP Authentication Types

- For all authentication types, except **ClientCertificateAuthentication**, the transparent proxy enriches the request by adding the necessary `Authorization` header to the request (extracting the `authTokens` section from the Destination service response).
For more information on the fields in `authTokens`, see ["Find Destination" Response Structure \[page 259\]](#).
- For the authentication types **BasicAuthentication**, **OAuth2TechnicalUserPropagation**, **OAuth2ClientCredentials**, and **OAuth2Password**, no additional `Authorization` headers have to be provided to the transparent proxy as all necessary data for authorization is present in the destination.

- For the authentication types **OAuth2UserTokenExchange**, **OAuth2JWTBearer**, and **OAuth2SAMLBearerAssertion**, the cloud user identity must be passed to the transparent proxy as a token represented by a JSON Web token (JWT) via the `Authorization` header of scheme `Bearer`. The token is passed as an `x-User-Token` to the Destination service, which will return `authTokens`. They are processed by the transparent proxy and attached to the HTTP request as `Authorization` header.
- For authentication type **OAuth2AuthorizationCode**, the code from the authorization service must be passed to the transparent proxy via `Authorization` header of scheme `Bearer`. You can also pass the optional headers `x-redirect-uri` and `x-code-verifier` to the request. The same can be configured in the destination.
If any or both of the headers are present in the request as well as in the destination configuration, the transparent proxy uses the request headers with priority.
For more information, see [OAuth Authorization Code Authentication \[page 139\]](#).
- For authentication type **OAuth2RefreshToken**, a refresh token must be passed to the transparent proxy via `Authorization` header of scheme `Bearer`.
For more information, see [OAuth Refresh Token Authentication \[page 134\]](#).

Client Assertion

Client assertion OAuth flows are supported, and the transparent proxy propagates the required headers to the Destination service. For more information, see: [Using Client Assertion with OAuth Flows \[page 152\]](#).

ClientCertificateAuthentication

Note

For general information, see [ClientCertificateAuthentication \[page 103\]](#).

For this authentication type, the transparent proxy manages the client certificate addition (client certificates and CA certificates) by getting them from the destination and adding them to the HTTP client that is used to establish the communication between client and server.

Limitations for usage:

- Supported cert store types are `p12`, `pfx`, `pem`, `der`, `cer`, `crt`.
- Only `RSA_pkcs1` keys (starting with header `RSA PRIVATE KEY` and having headers with encryption information) are currently supported for decryption from `pem` format. The following encryption algorithms are supported:
 - DES-CBC
 - DES-EDE3-CBC
 - AES-128-CBC
 - AES-192-CBC
 - AES-256-CBC
 All key types should be supported if no encryption is present.
- `KeyStore.Source=ClientProvided` is not supported as the transparent proxy can manage only certificates stored in the Destination service.

Principal Propagation

Principal propagation, also known as user propagation, lets you perform single sign-on (SSO) of the cloud user towards an on-premise system.

The cloud user's identity is passed to the transparent proxy as a token represented by a JSON Web token (JWT) via `Authorization` header of scheme `Bearer`. It is forwarded via the transparent proxy to the connectivity proxy and then to the Cloud Connector, which validates and further processes it to establish SSO with the on-premise system. The user is propagated via `SAP-Connectivity-Authentication` header.

For more information, see [Authenticating Users against On-Premise Systems \[page 419\]](#) and [Set Up Trust \[page 420\]](#).

Managed Namespaces Mode

The transparent proxy can operate in all namespaces in the cluster or only in namespaces labeled in the following way: `transparent-proxy.connectivity.api.sap/namespace:<namespace where Transparent Proxy is installed>`.

For this feature, you can set the Helm property `config.managedNamespacesMode` to "all" or "labelSelector". The default value is "all". It means that the proxy operates with destination custom resources across all namespaces in the cluster.

For more information, see [Configuration Guide \[page 822\]](#).

1.4.1.5 Managed Namespaces Mode

Manage namespaces for the transparent proxy for Kubernetes.

The transparent proxy can operate in all namespaces in the cluster, or only in namespaces labeled in the following way: `transparent-proxy.connectivity.api.sap/namespace:<namespace where Transparent Proxy is installed>`.

For this feature, you can set the Helm property `config.managedNamespacesMode` to **all** or **labelSelector**.

The default value is **all**. This means that the proxy operates with destination custom resources across all namespaces in the cluster.

For more information, see [Configuration Guide \[page 822\]](#).

1.4.2 Multitenancy

Use multitenancy for the transparent proxy for Kubernetes.

The transparent proxy works in two tenant modes: *dedicated* and *shared*. You can change the mode by modifying the Helm property `.config.tenantMode`.

For more information, see [Configuration Guide \[page 822\]](#).

- In the *dedicated* tenant mode, the transparent proxy can expose destinations only from the subaccount passed in the service key of the Destinations service. Requests to the transparent proxy remain unchanged.
- In the *shared* tenant mode, the transparent proxy can expose destinations from any subscribed tenant to the subaccount passed in the service key of the Destinations service.

HTTP

When requesting a destination through the transparent proxy in shared tenant mode, you must pass the `X-Tenant-Subdomain: <tenant-subdomain>` or `X-Tenant-Id: <tenant-id>` header.

A tenant is successfully onboarded when it is visible in the destination custom resource status. The onboarding of a tenant is dynamic and happens during the first request on behalf of that tenant.

TCP

Tenants are configured in the destination custom resource (CR) as annotation with key "transparent-proxy.connectivity.api.sap/tenant-subdomains" and value <all tenant subdomains> in form of a json array:

```
"transparent-proxy.connectivity.api.sap/tenant-subdomains":
['tenantSubdomain1', 'tenantSubdomain2', ...]'
```

For each tenant, a new instance of TCP proxy is created which is assigned only to that specific tenant and destination. Also, a separate service is created with a prefix containing the tenant subdomain (for example, if we have tenant subdomain "tenant1" and destination CR "dest", the service name will be "tenant1-dest").

Each tenant is successfully onboarded when it is visible in a Destination CR status.

1.4.3 Destination Custom Resource Scope

Define the access control scope of the destination custom resources for the transparent proxy for Kubernetes.

The scoping of access to destinations exposed via *destination custom resources* is based on the *network policies* concept in Kubernetes.

Prerequisites

Network policies are implemented by a CNI Plugin. To apply network policies, you must use a networking solution that supports the *NetworkPolicy* Kubernetes resource.

Note

Creating a *NetworkPolicy* resource without having a corresponding controller running in the cluster, will have **no effect**. In this case, access scoping will not work.

For more information, see [Network Policies](#).

Procedure

You can set a default scope on all destination custom resources by setting `.Values.config.security.accessControl.destinations.defaultScope` in the `values.yaml` file.

- If `.Values.config.security.accessControl.destinations.defaultScope` is set to **namespaced**, the destinations exposed via destination custom resources that were created in a namespace, are accessible only by the applications running in this namespace of the destination custom resource.
- If `.Values.config.security.accessControl.destinations.defaultScope` is set to **clusterWide**, the destinations exposed via destination custom resources are accessible from any namespace in the cluster.
- The default value is **namespaced**.

1.4.4 Integration with SAP BTP Connectivity

Integrate the transparent proxy with other SAP BTP Connectivity services.

[Destination Service Integration \[page 817\]](#)

Integrate the transparent proxy for Kubernetes with the SAP BTP Destination service.

[Connectivity Proxy Integration \[page 818\]](#)

Integrate the transparent proxy with the connectivity proxy for Kubernetes.

1.4.4.1 Destination Service Integration

Integrate the transparent proxy for Kubernetes with the SAP BTP Destination service.

The Destination service lets you declaratively model a technical connection as a destination configuration, and via its REST APIs find the destination configuration that is required to access a remote service or system from your Kubernetes workload. To integrate the transparent proxy with the Destination service, you must:

1. Create a Destination service instance or reuse an existing one.
2. Create a service key of the service instance.
3. Follow the instructions under [Lifecycle Management \[page 819\]](#) and set up the `values.yaml` for pairing with the Destination service, providing a base64-encoded service key for the Destination service.
4. Check the [Configuration Guide \[page 822\]](#) for the `config.integration.destinationService.instances` property.

1.4.4.2 Connectivity Proxy Integration

Integrate the transparent proxy with the connectivity proxy for Kubernetes.

The connectivity proxy is a Kubernetes component that connects workloads running on a Kubernetes cluster to on-premise systems exposed via the [Cloud Connector \[page 343\]](#). The transparent proxy itself doesn't provide direct connections to on-premise systems, that is, the connectivity proxy integration is mandatory for the transparent proxy to route to on-premise destinations. To integrate the transparent proxy with the connectivity proxy, you must:

1. Install a connectivity proxy instance or reuse an existing one. It must be installed in the same Kubernetes cluster where the transparent proxy is installed.
For more information, see [Lifecycle Management \[page 764\]](#).
2. Follow the instructions under [Lifecycle Management \[page 819\]](#) and set up the `values.yaml` for integrating with the connectivity proxy, providing the `config.integration.connectivityProxy.serviceName`.
For more information, see [Configuration Guide \[page 822\]](#).
3. If the connectivity proxy runs in multi-region mode, you can link a transparent proxy configuration for a Destination service instance with a connectivity proxy local region configuration. This can be done at design time by adding an association. Doing this, you won't need to provide the HTTP header `SAP-Connectivity-Region-Configuration-Id` on each request - the transparent proxy will automatically pass it to the connectivity proxy:


```
config:
  integration:
    destinationService:
      instances:
      - name: <local-instance-name>
        serviceCredentials:
          secretKey: <secret-key>
          secretName: <secret-name>
          secretNamespace: <secret-namespace>
        associateWith:
          connectivityProxy:
            locallyConfiguredRegionId: <locally-configured-conn-proxy-region-id>
```

Related Information

[Connectivity Proxy for Kubernetes \[page 732\]](#)

1.4.5 Lifecycle Management

Use the Helm chart to configure and manage the lifecycle of the transparent proxy.

The transparent proxy delivery includes a Helm chart that you can use for lifecycle management. The Helm allows full configuration via [the standard Helm method of a "values.yaml" file](#) .

The transparent proxy Helm chart is available via the RBSC (*repository-based shipment channel*) Helm repository and DockerHub.

Latest Helm Chart Version

The latest helm chart version is 1.5.0 for both RBSC and DockerHub.

RBSC Pull information (Latest Version)

Registry: 73554900100900006891.helmsrv.cdn.repositories.cloud.sap

Tag: 1.5.0

Authorization: See [RBSC documentation](#).

```
helm repo add transparent-proxy-helm https://
73554900100900006891.helmsrv.cdn.repositories.cloud.sap --username <user> --
password <pass>
helm pull transparent-proxy-helm/transparent-proxy --version=<version of helm
chart>
helm repo update
```

Deploy (DockerHub)

To deploy the transparent proxy on a Kubernetes cluster, execute the following steps:

1. Add the Helm repository and pull the Helm chart from DockerHub.

You can directly install it with one command:

```
helm install transparent-proxy oci://registry-1.docker.io/sapse/transparent-proxy --version <version of helm chart> --namespace <namespace> -f <path-to-values.yaml>
```

For additional information about using OCI registries with Helm, see [Helm docs](#).

2. (Optional) Prepare the Docker image configuration for the repository authentication if want to use a registry different from *docker.io*:
 - Create docker secret:

```
kubectl create secret docker-registry docker-secret --docker-server=73554900100900006891.dockersrv.cdn.repositories.cloud.sap --docker-username=<rbsc-user> --docker-password=<password>
```

- Place the name of secret it in the `values.yaml` for the `deployment.image.pullSecret` property. For more information, see [Configuration Guide \[page 822\]](#).
3. Prepare credentials for the Destination service instance :
 - Option 1: Reuse existing secret
 - If you already have an existing secret in your Kubernetes cluster you can reference it providing the parameter `config.integration.destinationService.instances[n].serviceCredentials.secretName`.
 - The secret data format can be either one-key or multi-key (for example, secrets created by Destination service instance creation in the Kyma dashboard).
 - If the secret data key format is one-key you should also provide the value of that key as `config.integration.destinationService.instances[n].serviceCredentials.secretKey`.
 - If that secret is not in the same namespace as the transparent proxy provide also `config.integration.destinationService.instances[n].serviceCredentials.secretNamespace`.
 - Option 2: Create secret holding the Destination service key
 - Get the service key of the Destination service instance.
 - Base64-encode the service key and place it in the `values.yaml` for `config.integration.destinationService.instances[n].serviceCredentials.secretData`.

For more information, see [Configuration Guide \[page 822\]](#).

Note

For x.509-based service keys with self-signed certificates, prepare a Kubernetes secret holding the private key for the certificate.

Example:

```
kubectl create secret generic x509-svc-key-private-key --from-file=pk.pem
```

If you are using another secret name or internal key in the secret, provide the required parameters (`config.integration.destinationService.instances[n].serviceCredentials.privateKey.secretName` and

`config.integration.destinationService.instances[n].serviceCredentials.privateKey.secretInternalKey`) in the `values.yaml`.

For more information, see [Configuration Guide \[page 822\]](#).

4. Fill all other properties in `values.yaml` for your scenario, as described in [Configuration Guide \[page 822\]](#).

⚠ Caution

Do not extract the archive and modify the default `values.yaml` file included there. Instead, use your own `values.yaml` and only include fields that should be overridden.

1.

📌 Note

By default, mTLS encryption is enabled using `cert-manager`. You should check the [Configuration Guide \[page 822\]](#) to modify the configuration settings according to your setup, for example, referencing your `cert-manager Issuer` or `ClusterIssuer`.

→ Tip

Encryption for the transparent proxy components can also be disabled for test purposes or if you have implemented your own mTLS sidecar solution, for example, Istio.

5. Use the Helm CLI to deploy the transparent proxy:

```
helm install transparent-proxy transparent-proxy-helm/transparent-proxy --  
version=<version of helm chart> --namespace <namespace> -f <path-to-  
values.yaml>
```

📌 Note

Installation in a Kyma Cluster

In case of installation in a Kyma cluster, if you want to enable istio for the transparent proxy workloads, you need to label the namespace where the transparent proxy is installed with:

```
kubectl label namespace <namespace> istio-injection=enabled
```

Update / Upgrade / Downgrade

⚠ Caution

Changing any of the `values.yaml` configurations using `helm upgrade` may result in the restart of some or all transparent proxy components.

When you have a transparent proxy deployed on the cluster, you may want to maintain it by changing configurations and/or changing its version. To do so, follow these steps:

1. Get the transparent proxy Helm chart. It can be the same version as the one currently installed or a different version that you want to upgrade or downgrade.
2. Prepare the `values.yaml` file for your scenario, as described in [Configuration \[page 823\]](#). Here you can just modify the one you used previously by applying the changes you desire.
3. Use the Helm CLI to upgrade the transparent proxy:

```
helm upgrade transparent-proxy oci://registry-1.docker.io/sapse/transparent-proxy --version <version of helm chart> --namespace <namespace> -f <path-to-new-values.yaml>
```

Undeploy

If you need to remove the transparent proxy from your cluster or from a namespace, you can delete all resources installed by Helm except for the [Custom Resource Definition](#) (CRD) of type `destinations.destination.connectivity.api.sap` using the Helm CLI:

```
helm uninstall transparent-proxy
```

If you need to remove the CRD of type `destinations.destination.connectivity.api.sap`, you can do it manually using [kubect](#).

```
kubectl delete crd destinations.destination.connectivity.api.sap
```

This will delete the CRD `destinations.destination.connectivity.api.sap` as well as all custom resources of type `destination.connectivity.api.sap` in the cluster.

Deploy/Update/Undeploy using Landscaper

To deploy/update/undeploy the transparent proxy on a Kubernetes cluster via Landscaper, look at the following [Guided Tour with Landscaper](#). You will need an LAAS instance that you can request from the corresponding team, or launch a local one using the Landscaper CLI.

1.4.5.1 Configuration Guide

Use the parameters below to configure the transparent proxy for Kubernetes.

⚠ Caution

Changing any of the properties below using *helm upgrade* may result in the restart of some or all transparent proxy components.

📌 Note

[n] in the table below means that the described property is part of an array.

Configuration

Parameter	Description	Default Value	Example	Required
<code>config.tenantMode</code>	<p>Defines the tenant mode in which the transparent proxy is working.</p> <p>Default value "<i>dedicated</i>" specifies that only destinations defined in the subaccount in a service key can be exposed and accessed via the transparent proxy. The other option "<i>shared</i>" specifies that the proxy can work with different tenants subscribed to the provider in the service key. In this mode, on each request the "<i>X-Tenant-Subdomain</i>" or "<i>X-Tenant-Id</i>" header is required. The tenant mode "<i>shared</i>" works only for destinations of type HTTP.</p>	"dedicated"		
<code>config.integration.destinationService.connectionTimeoutSeconds</code>	Connect timeout that would be used when getting access tokens and Destination service clients. [1, 60] seconds allowed.	5	20	False

Parameter	Description	Default Value	Example	Required
<code>config.integration.destinationService.readTimeoutSeconds</code>	Read timeout that would be used when getting access tokens and Destination service clients. [1, 60] seconds allowed.	10	20	False
<code>config.integration.connectivityProxy.connectionTimeoutSeconds</code>	The connect timeout must be used when accessing an HTTP system with <code>ProxyType=OnPremise</code> through the connectivity proxy. [1, 60] seconds allowed.	1	20	False
<code>config.integration.connectivityProxy.serviceName</code>	Kubernetes service name and namespace associated with the connectivity proxy workload.	None	<code>connectivity-proxy.<namespace></code>	False
<code>config.integration.connectivityProxy.http.port</code>	Port on which the HTTP interface of the connectivity proxy is started.	20003		False
<code>config.integration.connectivityProxy.tcp.port</code>	Port on which the TCP interface of the connectivity proxy is started.	20004		False
<code>config.integration.serviceMesh.istio.istio-injection</code>	Integration with Istio service mesh. The only acceptable value is 'enabled'. If set, the transparent proxy components will be integrated in the mesh, otherwise they will be exclusively removed from the mesh. If the configuration is changed, it triggers a restart.	"enabled"		False
<code>config.logging.level</code>	The initial log level across all transparent proxy components. The log levels can be changed dynamically, see Troubleshooting .	info		False

Parameter	Description	Default Value	Example	Required
<code>config.managedNamespacesMode</code>	<p>The mode in which transparent proxy operates with the destinations in the cluster. Possible values are "all" or "labelSelector".</p> <p>If "labelSelector" is set, the transparent proxy operates only in namespaces labeled with <i>transparent-proxy.connectivity.api.sap/namespace:<namespace where TP is installed in></i>.</p>	all		False
<code>config.manager.executionIntervalMinutes</code>	<p>The execution interval in minutes, on which the destination resource updates are fetched and applied to the cluster.</p> <div style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 10px; margin-top: 10px;"> <p>→ Tip</p> <p>We recommend that you do not set this value higher than 5, if there are frequent updates on destinations.</p> </div>	3		False
<code>config.security.accessControl.destinations.defaultScope</code>	<p>Defines the default scope of destination custom resources. Possible values are "namespaced" or "clusterWide".</p> <p>See Destination Custom Resource Scope [page 816].</p>	"namespaced"		True

Parameter	Description	Default Value	Example	Required
<code>config.security.communication.internal.encryptionEnabled</code>	<p>Enables/disables mTLS communication between the transparent proxy components.</p> <p>It should be disabled only in test environments or if you implement your own mTLS solution like Istio for example.</p> <div style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>Make sure you install the cert-manager in advance.</p> </div>	true		
<code>config.security.communication.internal.certManager issuerRef.kind</code>	<p>The <i>cert-manager</i> custom Kubernetes resource type of the CA. Must be one of (<i>Issuer</i> or <i>ClusterIssuer</i>).</p> <p>Only applicable for cert-manager.io (https://cert-manager.io/docs/).</p>		Issuer	False when encryption is disabled
<code>config.security.communication.internal.certManager issuerRef.name</code>	<p>The name of the installed cert-manager issuer.</p>		sap-transp-proxy-issuer	False when encryption is disabled
<code>config.security.communication.internal.certManager issuerRef.namespace</code>	<p>The namespace of the installed cert-manager issuer. Only applicable for cert.gardener.cloud (https://github.com/gardener/cert-management).</p>			

Parameter	Description	Default Value	Example	Required
<code>config.security.communication.internal.certManager.certificate.privateKey.algorithm</code>	Check cert-manager private key docs . Only applicable for cert-manager.io (https://cert-manager.io/docs/).		ECDSA	False
<code>config.security.communication.internal.certManager.certificate.privateKey.encoding</code>	Check cert-manager private key docs . Only applicable for cert-manager.io (https://cert-manager.io/docs/).		PKCS8	False
<code>config.security.communication.internal.certManager.certificate.privateKey.size</code>	Check cert-manager private key docs . Only applicable for cert-manager.io (https://cert-manager.io/docs/).		256	False
<code>config.security.communication.internal.certManager.certificate.duration</code>	The duration for which the certificates will be valid. Minimum accepted duration is 1 hour. Value must be in units accepted by Go time.ParseDuration . Only applicable for cert-manager.io (https://cert-manager.io/docs/).		720h	False
<code>config.security.communication.internal.certManager.certificate.renewBefore</code>	How long before expiry the <i>cert-manager</i> should renew the currently issued certificate. Value must be in units accepted by Go time.ParseDuration . Only applicable for cert-manager.io (https://cert-manager.io/docs/).		240h	False

Parameter	Description	Default Value	Example	Required
<code>config.integration.destinationService.defaultInstanceName</code>	Name of the default destination service instance to be used when <code>destinationServiceInstanceName</code> is not provided in the destination custom resource specification.			False
<code>config.integration.destinationService.instances[n].name</code>	Name of the destination service instance that is described. You can use it as <code>config.integration.destinationService.defaultInstanceName</code> or reference it in the destination custom resource specification as <code>destinationServiceInstanceName</code> .			False
<code>config.integration.destinationService.instances[n].serviceCredentials.secretName</code>	The name of the existing secret, which holds the credentials for the Destination service or the name of the secret to be created based on the "secretName", "secretData", "secretKey" fields.	<code>destination-service-key</code>		False

Parameter	Description	Default Value	Example	Required
<code>config.integration.destinationService.instances[n].serviceCredentials.secretKey</code>	<p>The key in the Destination service secret resource, which holds the base64-encoded value of the destination service key.</p> <div style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>Make sure you provide the right key for an existing secret. Required when the transparent proxy should create a secret and not required when describing an existing secret.</p> </div>			False
<code>config.integration.destinationService.instances[n].serviceCredentials.secretData</code>	The base64-encoded value of the service key, obtained from the Destination service instance. Required when transparent proxy should create secret and not required when describing an existing secret.			False
<code>config.integration.destinationService.instances[n].serviceCredentials.secretNamespace</code>	The namespace of the existing secret to be used, which holds the credentials for the Destination service. This field should not be present if there is no existing secret and the transparent proxy would create one based on the "secretName", "secretData", "secretKey" fields.			False
<code>config.integration.destinationService.instances[n].serviceCredentials.privateKey.secretName</code>	The name of the Kubernetes secret, which holds the private key to authenticate if using an x.509-based service key to the Destination service.	x509-svc-key-private-key		False

Parameter	Description	Default Value	Example	Required
<code>config.integration.destinationService.instances[n].serviceCredentials.privateKey.secretInternalKey</code>	The name of the internal key inside the Kubernetes secret holding the private key to authenticate if using an x.509-based service key to the Destination service.	pk.pem		False
<code>config.integration.destinationService.instances[n].associateWith.connectivityProxy.locallyConfiguredRegionId</code>	The locally configured region id from the connectivity proxy multi region configurations, that will be used by default when consuming destinations from the given destination service instance. Will be overridden for HTTP scenarios if header <code>SAP-Connectivity-Region-Configuration-Id</code> is passed.			False
<code>deployment.image.registry</code>	Name of the registry from which the transparent proxy will be downloaded.	docker.io		False
<code>deployment.image.repository</code>	Name of the repository.	sapse		False
<code>deployment.image.pullPolicy</code>	One of Always, Never, IfNotPresent. For more information, see Updating Images (Kubernetes documentation).	"IfNotPresent"		False
<code>deployment.image.pullSecret</code>	The secret used for authenticating against the repository (not required when using the Docker registry).			False
<code>deployment.image.tag</code>	Version of the transparent proxy images to be deployed.	<Version of the helm chart>	1.0.0	False
<code>deployment.replicas.http</code>	Amount of transparent HTTP proxy pods to start.	1		False

Parameter	Description	Default Value	Example	Required
<code>deployment.rePLICAS.tcp</code>	Amount of transparent TCP proxy pods for each TCP destination to start from this deployment.	1		False
<code>deployment.resources.http.request.cpu</code>	The Kubernetes scheduler uses this information to decide which node to place the pod on. If there are no nodes with the specified amount of CPU resources, the pod won't be scheduled (and therefore started).	0.2		False
<code>deployment.resources.http.request.memory</code>	The Kubernetes scheduler uses this information to decide which node to place the pod on. If there are no nodes with the specified amount of memory, the pod won't be scheduled (and therefore started).	256M		False
<code>deployment.resources.http.limits.cpu</code>	The Kubelet enforces the limit so that the running container is not allowed to use more CPU resources than the set limit. If the limit is crossed, the process is throttled.	0.4		False
<code>deployment.resources.http.limits.memory</code>	The Kubelet enforces the limit so that the running container is not allowed to use more memory than the set limit. If the limit is crossed, the process is terminated with an out of memory (OOM) error.	512M		False



Parameter	Description	Default Value	Example	Required
<code>deployment.resources.tcp.request.cpu</code>	The Kubernetes scheduler uses this information to decide which node to place the pod on. If there are no nodes with the specified amount of CPU resources, the pod won't be scheduled (and therefore started).	0.05		False
<code>deployment.resources.tcp.request.memory</code>	The Kubernetes scheduler uses this information to decide which node to place the Pod on. If there are no nodes with the specified amount of memory, the pod won't be scheduled (and therefore started).	32M		False
<code>deployment.resources.tcp.limits.cpu</code>	The Kubernetes enforces the limit so that the running container is not allowed to use more CPU resources than the set limit. In case the limit is crossed, the process would be throttled.	0.1		False
<code>deployment.resources.tcp.limits.memory</code>	The Kubelet enforces the limit so that the running container is not allowed to use more memory than the set limit. In case the limit is crossed, the process would be terminated with an out-of-memory (OOM) error.	64M		False

Parameter	Description	Default Value	Example	Required
<code>deployment.autoscaling.http.horizontal.enabled</code>	Enables or disables the <i>Horizontal Pod Autoscaler</i> mechanism see Horizontal Pod Autoscaler (Kubernetes documentation) for HTTP proxy pods.	False		False
<div style="border-left: 2px solid #0070C0; padding-left: 10px; background-color: #F0F0F0;"> <p>Note</p> <p>You cannot enable both horizontal and vertical autoscaling. If you want to use horizontal autoscaling, <code>deployment.autoscaling.http.vertical.enabled</code> must not be set to true.</p> </div>				
<code>deployment.autoscaling.http.horizontal.maxReplicaCount</code>	Upper limit for the number of HTTP transparent proxy replicas to which the autoscaler can scale up. It should be higher than <code>deployment.replicas.http</code> .	2		False
<code>deployment.autoscaling.http.horizontal.metrics.cpuAverageUtilization</code>	Target value of the average CPU metric across all transparent HTTP proxy pods, represented as a percentage of the requested value of the CPU for the pods.	80		False
<code>deployment.autoscaling.http.horizontal.metrics.memoryAverageUtilization</code>	Target value of the average memory metric across all transparent HTTP proxy pods, represented as a percentage of the requested value of the memory for the pods.	80		False

Parameter	Description	Default Value	Example	Required
<code>deployment.autoscaling.http.vertical.enabled</code>	<p>Enables or disables the <i>Vertical Pod Autoscaler</i> mechanism for http proxy pods.</p> <p>To take effect, the vertical pod autoscaling mechanism for the cluster should be enabled.</p> <p>See Vertical Pod Autoscaling for details.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Note</p> <p>You cannot enable both horizontal and vertical autoscaling. If you want to use vertical autoscaling, <code>deployment.autoscaling.http.horizontal.enabled</code> must not be set to true.</p> </div>	false		False
<code>deployment.autoscaling.http.vertical.updateMode</code>	<p>The mode in which the <i>Vertical Pod Autoscaler</i> should operate.</p> <p>See Vertical Pod Autoscaler Quick start for details.</p>	"Off"		False
<code>deployment.autoscaling.http.vertical.minAllowed.cpu</code>	<p>Specifies the minimal amount of CPU that will be recommended for each HTTP proxy pod. By default there is no minimum.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Note</p> <p>There are 2 containers in each HTTP proxy pod.</p> </div>		0.2	False

Parameter	Description	Default Value	Example	Required
<code>deployment.autoScaling.http.vegetical.minAllowed.memory</code>	Specifies the minimal amount of memory that will be recommended for each HTTP proxy pod. By default, there is no minimum.		256M	False
<div style="background-color: #f0f0f0; padding: 10px; border-left: 2px solid #0070c0;"> <p>Note</p> <p>There are 2 containers in each HTTP proxy pod.</p> </div>				
<code>deployment.autoScaling.http.vegetical.maxAllowed.cpu</code>	Specifies the maximum amount of CPU that will be recommended for each HTTP proxy pod. By default, there is no maximum.		0.4	False
<div style="background-color: #f0f0f0; padding: 10px; border-left: 2px solid #0070c0;"> <p>Note</p> <p>There are 2 containers in each HTTP proxy pod.</p> </div>				
<code>deployment.autoScaling.http.vegetical.maxAllowed.memory</code>	Specifies the maximum amount of memory that will be recommended for each HTTP proxy pod. By default, there is no maximum.		512M	False
<div style="background-color: #f0f0f0; padding: 10px; border-left: 2px solid #0070c0;"> <p>Note</p> <p>There are 2 containers in each HTTP proxy pod.</p> </div>				

Parameter	Description	Default Value	Example	Required
<code>deployment.autoscaling.tcp.horizontal.enabled</code>	Enables or disables the <i>Horizontal Pod Autoscaler</i> mechanism see Horizontal Pod Autoscaler (Kubernetes documentation) for TCP proxy pods.	False		False
<div style="border-left: 2px solid #0070C0; padding-left: 10px; background-color: #F0F0F0;"> <p>Note</p> <p>You cannot enable both horizontal and vertical autoscaling. If you want to use horizontal autoscaling, <code>deployment.autoscaling.tcp.vertical.enabled</code> must not be set to true.</p> </div>				
<code>deployment.autoscaling.tcp.horizontal.maxReplicaCount</code>	Upper limit for the number of TCP transparent proxy replicas to which the autoscaler can scale up. It should be higher than <code>deployment.replicas.tcp</code> .	2		False
<code>deployment.autoscaling.tcp.horizontal.metrics.cpuAverageUtilization</code>	Target value of the average CPU metric across all transparent TCP proxy pods for a given destination instance, represented as a percentage of the requested value of the CPU for the pods.	80		False
<code>deployment.autoscaling.tcp.horizontal.metrics.memoryAverageUtilization</code>	Target value of the average memory metric across transparent TCP proxy pods for a given destination instance, represented as a percentage of the requested value of the memory for the pods.	80		False

Parameter	Description	Default Value	Example	Required
<code>deployment.autoscaling.tcp.vertical.enabled</code>	<p>Enables or disables the <i>Vertical Pod Autoscaler</i> mechanism for TCP proxy pods.</p> <p>To take effect, the vertical pod autoscaling mechanism for the cluster should be enabled.</p> <p>See Vertical Pod Auto-Scaling  for details.</p> <div style="border: 1px solid #ccc; background-color: #f9f9f9; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>You cannot enable both horizontal and vertical autoscaling. If you want to use vertical autoscaling, <code>deployment.autoscaling.tcp.horizontal.enabled</code> must not be set to true.</p> </div>	false		False
<code>deployment.autoscaling.tcp.vertical.updateMode</code>	<p>The mode in which the <i>Vertical Pod Autoscaler</i> should operate.</p> <p>See Vertical Pod Autoscaler Quick start  for details.</p>	"Off"		False
<code>deployment.autoscaling.tcp.vertical.minAllowed.cpu</code>	<p>Specifies the minimal amount of CPU that will be recommended for each TCP proxy pod.</p> <p>By default there is no minimum.</p>		0.2	False

Parameter	Description	Default Value	Example	Required
<code>deployment.autoScaling.tcp.vertical.minAllowed.memory</code>	Specifies the minimal amount of memory that will be recommended for each TCP proxy pod. By default there is no minimum.		256M	False
<code>deployment.autoScaling.tcp.vertical.maxAllowed.cpu</code>	Specifies the maximum amount of CPU that will be recommended for each TCP proxy pod. By default there is no maximum.		0.4	False
<code>deployment.autoScaling.tcp.vertical.maxAllowed.memory</code>	Specifies the maximum amount of memory that will be recommended for each TCP proxy pod. By default there is no maximum.		512M	False

If you want to pull docker images from the *SAP Repository-Based Shipment Channel (RBSC)*, use the following configuration:

- `deployment.image.registry = 73554900100900006891.dockersrv.cdn.repositories.cloud.sap`
- `deployment.image.repository = com.sap.core.connectivity.transparent-proxy`
- `deployment.image.pullSecret = See Lifecycle Management \[page 819\].`

1.4.5.2 Sizing Recommendations

When installing a transparent proxy for Kubernetes, the first thing you need to decide is the sizing of the installation.

Overview

To get the most out of the transparent proxy, you must configure it in a suitable way, fitting the scenarios in which it plays a role.

Sizing Options

The following table gives basic sizing guidance. The values listed in the `CPU`, `Memory` and sections correspond to the properties, which should be defined in the `values.yaml`. For more information, see [Configuration Guide \[page 822\]](#):

HTTP Proxy Sizing

Size	CPU	Memory
S: The expected load is small - request concurrency and size are low	<ul style="list-style-type: none"> <code>deployment.resources.ht</code> <code>tp.requests.cpu: 0.2</code> <code>deployment.resources.ht</code> <code>tp.limits.cpu: 0.4</code> 	<ul style="list-style-type: none"> <code>deployment.resources.ht</code> <code>tp.requests.memory: 256 M</code> <code>deployment.resources.ht</code> <code>tp.limits.memory: 512 M</code>
M: The expected load is medium - request concurrency and size are medium	<ul style="list-style-type: none"> <code>deployment.resources.ht</code> <code>tp.requests.cpu: 0.4</code> <code>deployment.resources.ht</code> <code>tp.limits.cpu: 0.8</code> 	<ul style="list-style-type: none"> <code>deployment.resources.ht</code> <code>tp.requests.memory: 512 M</code> <code>deployment.resources.ht</code> <code>tp.limits.memory: 1024 M</code>
L: The expected load is large - request concurrency and size are medium or high	<ul style="list-style-type: none"> <code>deployment.resources.ht</code> <code>tp.requests.cpu: 0.8</code> <code>deployment.resources.ht</code> <code>tp.limits.cpu: 1.6</code> 	<ul style="list-style-type: none"> <code>deployment.resources.ht</code> <code>tp.requests.memory: 1024 M</code> <code>deployment.resources.ht</code> <code>tp.limits.memory: 2048 M</code>

TCP Proxy Sizing

Size	CPU	Memory
S: The expected load is small - request concurrency and size are low	<ul style="list-style-type: none"> <code>deployment.resources.tc</code> <code>p.requests.cpu: 0.05</code> <code>deployment.resources.tc</code> <code>p.limits.cpu: 0.1</code> 	<ul style="list-style-type: none"> <code>deployment.resources.tc</code> <code>p.requests.memory: 32 M</code> <code>deployment.resources.tc</code> <code>p.limits.memory: 64 M</code>
M: The expected load is medium - request concurrency and size are medium	<ul style="list-style-type: none"> <code>deployment.resources.tc</code> <code>p.requests.cpu: 0.1</code> <code>deployment.resources.tc</code> <code>p.limits.cpu: 0.2</code> 	<ul style="list-style-type: none"> <code>deployment.resources.tc</code> <code>p.requests.memory: 64 M</code> <code>deployment.resources.tc</code> <code>p.limits.memory: 128 M</code>
L: The expected load is large - request concurrency and size are medium or high	<ul style="list-style-type: none"> <code>deployment.resources.tc</code> <code>p.requests.cpu: 0.2</code> <code>deployment.resources.tc</code> <code>p.limits.cpu: 0.4</code> 	<ul style="list-style-type: none"> <code>deployment.resources.tc</code> <code>p.requests.memory: 128 M</code> <code>deployment.resources.tc</code> <code>p.limits.memory: 256 M</code>

📌 Note

Relation to Connectivity Proxy

The above-mentioned sizing recommendations for the TCP proxies are related to the connectivity proxy software component, which also acts as a tunnel server to the on-premise systems it connects to. This means that the connectivity proxy has to be properly sized as well, see [Sizing Recommendations \[page 784\]](#) (connectivity proxy for Kubernetes).

📌 Note

General Note

These sizing recommendations are just a direction point. There are many factors that affect the performance offered by the transparent proxy, related to the specifics of your concrete scenarios, expected regular and intermittent load, and so on.

1.4.6 Monitoring

Check the availability, status, and destination custom resources of the transparent proxy for Kubernetes.

Availability Monitoring

The availability check of the transparent proxy is the minimal verification that can be done to ensure that the transparent proxy is alive and working. This is done by a health check application deployed to the cluster during installation of the transparent proxy.

Status Check

The `/status` check performs the following :

- If there are HTTP custom resources defined → is the `sap-transp-proxy-http` alive and running?
- If there are TCP custom resources defined → are all the `sap-transp-proxy-tcp` alive and running?
- Is the `sap-transp-proxy-manager` alive and running?

The overall status provided by the `/status` endpoint is formed in the following way:

- if all used components are in `ok` status → overall status is `ok`.
- if some, but not all components are in `critical` status → overall status is `warning`.
- if all components are in `critical` status → overall status is `critical`.

Destination Custom Resources Check

The `/destinationCRs` check provides information for all destination custom resources in the namespace where the transparent proxy is installed.

To find an example, see [Verification and Testing \[page 858\]](#).

1.4.7 Troubleshooting

Find troubleshooting information for the transparent proxy for Kubernetes.

Get Logs of the Transparent Proxy Components

The transparent proxy consists of a transparent proxy manager, transparent HTTP proxy, and multiple transparent TCP proxy instances.

1. To get the logs of the transparent proxy manager, execute:

```
kubectl logs -l transparent-proxy.connectivity.api.sap/component=manager --tail=-1 -n <installation-namespace> > transparent-proxy-manager.log
```

2. To get the logs of the transparent HTTP proxy, execute:

```
kubectl logs -l transparent-proxy.connectivity.api.sap/component=http --all-containers --tail=-1 -n <installation-namespace> > transparent-http-proxy.log
```

3. To get the logs of the transparent TCP proxy instances, execute:

```
kubectl logs -l transparent-proxy.connectivity.api.sap/component=tcp --tail=-1 -n <installation-namespace> > transparent-tcp-proxy.log
```

4. To get the logs of the transparent proxy health check, execute:

```
kubectl logs -l transparent-proxy.connectivity.api.sap/component=healthcheck --tail=-1 -n <installation-namespace> > transparent-proxy-health-check.log
```

5. To get the logs of the transparent proxy operator (installed only when [Transparent Proxy is enabled as a Kyma Module in the Kyma environment \[page 860\]](#)) execute:

```
kubectl logs -l transparent-proxy.connectivity.api.sap/component=operator --tail=-1 -n <installation-namespace> > transparent-proxy-operator.log
```

Files created by the above commands:

- `transparent-proxy-manager.log`

- `transparent-http-proxy.log`
- `transparent-tcp-proxy.log`
- `transparent-proxy-health-check.log`
- `transparent-proxy-operator.log`

can be used for investigation purposes. For more information, see [Recommended Actions \[page 854\]](#).

Get Status of `destinations.destination.connectivity.api.sap` Custom Resources

```
kubectl describe destinations -n <installation-namespace>
```

Change Log Levels of the Transparent Proxy Components

When the default logging level is not sufficient for debugging the issue you are facing, you can change the log level to get more insight about the problem.

Changing a log level is done without any downtime and requires no restarts. All you need to do is invoke a simple command on the pod of a transparent proxy component where you need to gain more insight. Here are some examples:

1. To change the log level of the transparent proxy manager, execute:

```
kubectl exec <transparent proxy manager pod> -n <installation-namespace> -it
-- /etc/logging/change-log-level DEBUG
```

2. To change the log level of the transparent HTTP proxy, execute:

```
kubectl exec <transparent http proxy pod> -n <installation-namespace> -it
-- /etc/logging/change-log-level DEBUG
```

3. To change the log level of a transparent TCP proxy instance, execute:

```
kubectl exec <transparent tcp proxy instance pod> -n <installation-namespace>
-it -- /etc/logging/change-log-level DEBUG
```

4. To change the log level of the transparent proxy health check, execute:

```
kubectl exec <transparent proxy health check pod> -n <installation-namespace>
-it -- /etc/logging/change-log-level DEBUG
```

5. To change the log level of the transparent proxy operator (installed only when [Transparent Proxy is enabled as a Kyma Module in the Kyma environment \[page 860\]](#)) execute:

```
kubectl exec <transparent proxy operator pod> -n <installation-namespace> -it
-- /etc/logging/change-log-level DEBUG
```

Accepted log levels are: `trace`, `debug`, `info`, `warn`, `error`, `fatal`. The default log level across all transparent proxy components is `info`.

Basic CPU and Memory Monitoring

When using `kubectl`, you can monitor CPU and memory of the transparent proxy's pods out of the box:

```
kubectl top pods -n <installation-namespace> -l 'transparent-proxy.connectivity.api.sap/component in (http,manager,tcp)'
```

For more information on using `kubectl`, see [Interacting with running Pods](#).

Common Issues and Solutions

Issue	Solution
The call to the Kubernetes service referencing your destination returns 404.	Make sure the URL configured in the destination is valid. Wait for the next transparent proxy manager completion.
The call to the Kubernetes service referencing your destination returns response with status code 400. Check also 'x-error-message', 'x-error-origin' and 'x-internal-error-code' response headers for more concrete information.	Check your request headers or OAuth Configuration fields in the referenced Destination. Example for 'authorization' header of scheme - 'Authorization: Bearer <base64-encoded-token>'
The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value 'x-token-service-tenant' request header is missing. It is required when 'tokenServiceURLType' property in the destination is of type Common.'	Make sure you add a valid 'x-token-service-tenant' request header to each destination that has 'tokenServiceURLType' property of type Common.
The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value 'Cannot mix header 'x-client-assertion-destination-name' with headers 'x-client-assertion' and 'x-client-assertion-type'.'	Make sure you pass either 'x-client-assertion-destination-name' or 'x-client-assertion' and 'x-client-assertion-type' request headers and not the combination of all of them.
The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value 'Destination service returned unsuccessful status code. Check your related request headers.' and 'Destination service' value in the 'x-error-origin' header.	Make sure you pass a valid 'authorization' header value that is appropriate for the referenced destination authentication type.

Issue	Solution
The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value 'In multi-tenant operational mode, either 'x-tenant-id' or 'x-tenant-subdomain' header is mandatory to be provided.'	Make sure you pass either x-tenant-id or x-tenant-subdomain header in shared mode.
The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value 'Both 'x-tenant-id' and 'x-tenant-subdomain' headers are passed. Only one should be provided.'	Make sure you pass either x-tenant-id or x-tenant-subdomain header but not both at the same time in shared mode.
The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value "x-destination-name' header is missing but required for an endpoint exposing multiple destinations via the same Destination CR.'	In case the spec.destinationRef.name property is "*" the 'x-destination-name' header is mandatory provide reference to a valid destination.
The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value 'The value of the header 'x-destination-name' exceeds the maximum allowed length of 200'.	The provided header value is wrong since the Destination Service allow maximum length for a destination name to be 200 symbols.
The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value 'The value of the header 'x-destination-name' contains characters that are not allowed. Check the destination name restrictions in the Destination service documentation.'	The provided header value for 'x-destination-name' has invalid characters. Check the name restrictions in the Destination service documentation.
The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value 'The value of the header 'x-fragment-name' exceeds the maximum allowed length of 200'.	The provided header value is wrong since the Destination Service allow maximum length for a fragment name to be 200 symbols.
The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value 'The value of the header 'x-fragment-name' contains characters that are not allowed. Check the destination name restrictions in the Destination service documentation.'	The provided header value for 'x-destination-name' has invalid characters. Check the name restrictions in the Destination service documentation.
The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value 'The requested endpoint is bound to a concrete destination via the respective Destination CR. 'x-destination-name' header is not supported for this endpoint. Contact the Transparent Proxy Administrator.'	The Destination Custom Resource is bound to a specific destination via spec.destinationRef.name property. Either change the spec.destinationRef.name property in the CR to "*" or do not pass the 'x-destination-name' header.

Issue	Solution
<p>The call to the Kubernetes service referencing your destination returns response with status code 400 and has response header 'x-error-message' with value 'The requested endpoint is bound to a concrete destination via the respective Destination CR. 'x-fragment-name' header is not supported for this endpoint. Contact the Transparent Proxy Administrator.'</p>	<p>The Destination Custom Resource is bound to a specific destination via spec.destinationRef.name property. Either change the spec.destinationRef.name property in the CR to "*" or set spec.fragmentRef.name to the desired fragment.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 502 and has response header 'x-error-message' with value 'Destination '<name>' referred in Destination CR '<name>' is not found in tenant '<tenant-subdomain>'. Inspect the Destination CR or contact the Destination Administrator.' and 'Destination service' value in the 'x-error-origin' header.</p>	<p>Check if your referenced destination exists in the specified Destination Service instance or contact the Destination Administrator.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 502 and has response header 'x-error-message' with value 'Destination '<name>' with '<fragmentName>' referred in Destination CR '<name>' is not found in tenant '<tenant-subdomain>'. Inspect the Destination CR or contact the Destination Administrator.' and 'Destination service' value in the 'x-error-origin' header.</p>	<p>Check if your referenced destination/fragment exists in the specified Destination Service instance or contact the Destination Administrator.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 502 and has response header 'x-error-message' with value 'Received invalid configuration from Destination Service, requested via destination '<destinationName>'. and 'Destination service' value in the 'x-error-origin' header.</p>	<p>There could be an issue with the configuration fields of the destination. Contact the Destination Administrator.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 502 and has response header 'x-error-message' with value 'Received invalid configuration from Destination Service, requested via destination '<destinationName>' and fragment '<fragmentName>'. and 'Destination service' value in the 'x-error-origin' header.</p>	<p>There could be an issue with the configuration fields of the destination/fragment. Contact the Destination Administrator.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 502 and has response header 'x-error-message' with value 'Invalid OAuth configuration in destination '<name>' referred in Destination CR '<name>'. Contact the Destination Administrator.' and 'Destination service' value in the 'x-error-origin' header.</p>	<p>Check Auth Configuration fields or your request headers in the referenced destination.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 502 and has response header 'x-error-message' with value 'Invalid OAuth configuration in destination '<name>' with fragment '<fragmentName>' referred in Destination CR '<name>'. Contact the Destination Administrator.' and 'Destination service' value in the 'x-error-origin' header.</p>	<p>Check Auth Configuration fields or your request headers in the referenced destination/fragment.</p>

Issue	Solution
<p>The call to the Kubernetes service referencing your destination returns response with status code 502 and has response header 'x-error-message' with value 'On-Premise technical connectivity is not configured. There is no value provided for connectivity proxy service name.' and 'Transparent Proxy' value in the 'x-error-origin' header.</p>	<p>Make sure you provide correct Connectivity proxy service-Name in Helm Configurations.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 502, and has response header 'x-error-message' with value 'The Destination service instance name in the Destination CR '%s' is not configured as expected.' and 'Transparent Proxy' value in the 'x-error-origin' header.</p>	<p>Check if you pass a valid Destination Service instance in Destination CR or contact the Kubernetes cluster administrator to check the Helm configuration.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestination-NotFound".</p>	<p>Create a destination with the name "testdest". Wait for the next Transparent Proxy Manager completion.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice" and references fragment called "testfragment". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedFragment-NotFound".</p>	<p>Create a fragment with name "testfragment". Wait for the next Transparent Proxy Manager completion.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice" and references fragment called "testfragment". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedFragment-NotFound".</p>	<p>Create a fragment with name "testfragment". Wait for the next Transparent Proxy Manager completion.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestination-TypeNotSupported".</p>	<p>Update the destination with the name "testdest" Type field with supported values: "HTTP", "TCP". Wait for the next Transparent Proxy Manager completion.</p>

Issue	Solution
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has status "False" with the reason "ConfigurationError-ReferencedDestination-ProxyTypeNotSupported".</p>	<p>Update the destination with the name "testdest" ProxyType field with supported values: "OnPremise", "Internet". Wait for the next Transparent Proxy Manager completion.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice" and references fragment called "testfragment". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has status "False" with the reason "ConfigurationError-ReferencedDestinationWithFragment-ProxyTypeNotSupported".</p>	<p>Update the destination with the name "testdest" or the fragment with name "testfragment" ProxyType field with supported values: "OnPremise", "Internet". Wait for the next Transparent Proxy Manager completion.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestination-UrlNotSupported".</p>	<p>Update the destination with the name "testdest" URL field with supported values: Should not be empty; If type is HTTP, URL should start with "http://" or "https://" or without protocol. Wait for the next Transparent Proxy Manager completion.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice" and references fragment called "testfragment". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestinationWithFragment-UrlNotSupported".</p>	<p>Update the destination with the name "testdest" or the fragment with name "testfragment" URL field with supported values: Should not be empty; If type is HTTP, URL should start with "http://" or "https://" or without protocol. Wait for the next Transparent Proxy Manager completion.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The destination is of type TCP and ProxyType OnPremise and there is no k8s service with that name created. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestination-TcpAddressNotSupported".</p>	<p>Update the destination with the name "testdest" Address field with supported values: Should not be empty; It should start with "tcp://" or without protocol. Wait for the next Transparent Proxy Manager completion.</p>

Issue	Solution
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice" and references fragment called "testfragment". The destination is of type TCP and Proxy-Type OnPremise and there is no k8s service with that name created. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestinationWithFragment-TcpAddressNotSupported".</p>	<p>Update the destination with the name "testdest" or the fragment with name "testfragment" Address field with supported values: Should not be empty; It should start with "tcp://" or without protocol. Wait for the next Transparent Proxy Manager completion.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestination-LoopbackUrlNotSupported".</p>	<p>Update the destination with the name "testdest" URL field with supported values: Should not be empty; URL should not be loopback (loopback URLs are those, which contain exactly one of the following hosts: "localhost", "127.0.0.1", "::1", "0:0:0:0:0:0:1"); Wait for the next Transparent Proxy Manager completion.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice" and references fragment called "testfragment". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestinationWithFragment-LoopbackUrlNotSupported".</p>	<p>Update the destination with the name "testdest" or the fragment with name "testfragment" URL field with supported values: Should not be empty; URL should not be loopback (loopback URLs are those, which contain exactly one of the following hosts: "localhost", "127.0.0.1", "::1", "0:0:0:0:0:0:1"); Wait for the next Transparent Proxy Manager completion.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestination-KeystoreInvalid".</p>	<p>Update the destination with the name "testdest" KeyStore-Location field with supported values. Supported cert store types are p12, pfx, pem, der, cer, crt.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestination-TrustStoreInvalid".</p>	<p>Update the destination with the name "testdest" TrustStore-Location field with supported values. Supported cert store types are p12, pfx, pem, der, cer, crt.</p>

Issue	Solution
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestination-HttpUrlWithTrustStoreOrKeyStoreNotSupported".</p>	<p>Update the destination with the name "testdest" so that it has not URL that starts with "http://" and has TrustStoreLocation or KeyStoreLocation in the same time.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice" and references fragment called "testfragment". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ReferencedDestination-HttpUrl-WithTrustStoreOrKeyStoreNotSupported".</p>	<p>Update the destination with the name "testdest" or the fragment with name "testfragment" so that it has not URL that starts with "http://" and has TrustStoreLocation or KeyStoreLocation in the same time.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationFailed."</p>	<p>Wait for the next Transparent Proxy Manager completion. If the issue remains, see Recommended Actions [page 854].</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice" and is a dynamic custom resource since it refers spec.destinationRef.name to "*". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-FragmentSpecification-NotSupported" with message: "Technical connectivity is not configured. Static declaration of a fragment is not supported for dynamic destination custom resources."</p>	<p>Dynamic custom resource should not have statically declared fragments in spec.fragmentRef.name.</p> <p>Fragment name should be provided only through the "x-fragment-name" header.</p>
<p>The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The call to the "testservice" Kubernetes service returns an unsuccessful status code or the Kubernetes service does not exist. The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ServiceCreationFailed."</p>	<p>Wait for the next Transparent Proxy Manager completion. If the issue remains, see Recommended Actions [page 854].</p>
<p>The call to the Kubernetes service referencing your destination returns an unsuccessful status code or the Kubernetes service does not exist.</p>	<p>Wait for the next Transparent Proxy Manager completion. If the issue remains, see Recommended Actions [page 854].</p>

Issue	Solution
<p>The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-Duplicate-DestinationCustomResource". There is a destinations.destination.connectivity.api.sap custom resource with the same name in another namespace.</p>	<p>Change the name of your destinations.destination.connectivity.api.sap custom resource or remove the other one if you have control of it.</p>
<p>The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ProxyTenantModeMismatch" and message "Technical connectivity is not configured. Connectivity Proxy tenant mode \"dedicated\" is not compatible with Transparent Proxy tenant mode \"shared\"".</p>	<p>The Connectivity proxy should be configured with the same tenant mode as the Transparent proxy.</p>
<p>The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-ProxyTenantModeMismatch" and message "Technical connectivity is not configured. Connectivity Proxy tenant mode \"shared\" is not compatible with Transparent Proxy tenant mode \"dedicated\"".</p>	<p>The Connectivity proxy should be configured with the same tenant mode as the Transparent proxy.</p>
<p>The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-MultitenancyAnnotation-Invalid" and message "Technical connectivity is not configured. \"transparent-proxy.connectivity.api.sap/tenant-subdomains\" annotation is invalid".</p>	<p>Use proper json array formatting when specifying the TCP tenants in your "transparent-proxy.connectivity.api.sap/tenant-subdomains" annotation.</p>
<p>The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "FailedToApplyConfiguration" and message "Configuration failed for tenants: [\"Tenant1\", \"Tenant2\"...]" where Tenant1, Tenant2, etc. are tenant subdomains specified in the "transparent-proxy.connectivity.api.sap/tenant-subdomains" annotation.</p>	<p>Check the transparent proxy manager for error logs. The tenants Tenant1, Tenant2 could not be onboarded because either the specified destination is not found in the destination service for the given tenants, is not valid in the destinations service or creating k8s components for the given destination/tenant failed.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 500, response header 'x-error-message' with value "Tenant subdomain '<tenantSubdomain>' could not be determined as valid tenant for Destination service instance with name '<destinationServiceInstanceName>' as provided in the Transparent Proxy deployment configuration. Contact the Tenant administrator." response header 'x-error-origin' with value "XSUAA", response header 'x-internal-error-code' with value "404".</p>	<p>The passed tenant in x-tenant-subdomain is not found. Make sure you pass existing tenant in x-tenant-subdomain.</p>

Issue	Solution
<p>The call to the Kubernetes service referencing your destination returns response with status code 500, response header 'x-error-message' with value "Tenant id '<tenantSubdomain>' could not be determined as valid tenant for Destination service instance with name '<destinationServiceInstanceName>' as provided in the Transparent Proxy deployment configuration. Contact the Tenant administrator." response header 'x-error-origin' with value "XSUAA", response header 'x-internal-error-code' with value "404".</p>	<p>The passed tenant in x-tenant-id is not found. Make sure you pass existing tenant in x-tenant-id.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 500, response header 'x-error-message' with value "Tenant subdomain '<tenantSubdomain>' is not subscribed to respective provider application using Destination service instance with name '<destinationServiceInstanceName>', as provided in the Transparent Proxy deployment configuration, under provider tenant subdomain '<providerTenantSubdomain>'. Contact the Destination Administrator." response header 'x-error-origin' with value "XSUAA", response header 'x-internal-error-code' with value "401".</p>	<p>The passed tenant in x-tenant-subdomain is not subscribed to the provider. Make sure you pass subscribed to the provider tenant in x-tenant-subdomain.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 500, response header 'x-error-message' with value "Tenant id '<tenantId>' is not subscribed to respective provider application using Destination service instance with name '<destinationServiceInstanceName>', as provided in the Transparent Proxy deployment configuration, under provider tenant subdomain '<providerTenantSubdomain>'. Contact the Destination Administrator." response header 'x-error-origin' with value "XSUAA", response header 'x-internal-error-code' with value "401".</p>	<p>The passed tenant in x-tenant-id is not subscribed to the provider. Make sure you pass subscribed to the provider tenant in x-tenant-id.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 502, response header 'x-error-message' with value "Destination '<destinationName>' referred in Destination CR '<crName>' is not found in tenant '<tenantName>'. Inspect the Destination CR or contact the Destination Administrator." response header 'x-error-origin' with value "Destination Service", response header 'x-internal-error-code' with value "404".</p>	<p>The referenced destination is not found in the destination service for the given tenant.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 502, response header 'x-error-message' with value "Destination '<destinationName>' with fragment '<fragmentName>' referred in Destination CR '<crName>' is not found in tenant '<tenantName>'. Inspect the Destination CR or contact the Destination Administrator." response header 'x-error-origin' with value "Destination Service", response header 'x-internal-error-code' with value "404".</p>	<p>The referenced destination or fragment is not found in the destination service for the given tenant.</p>

Issue	Solution
<p>The call to the Kubernetes service referencing your destination returns response with status code 502, response header 'x-error-message' with value "Destination '<destinationName>' not found in tenant '<tenantName>'. Contact the Destination Administrator." response header 'x-error-origin' with value "Destination Service", response header 'x-internal-error-code' with value "404".</p>	<p>The referenced destination is not found in the destination service for the given tenant.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 502, response header 'x-error-message' with value "Destination '<destinationName>' with fragment '<fragmentName>' not found in tenant '<tenantName>'. Contact the Destination Administrator." response header 'x-error-origin' with value "Destination Service", response header 'x-internal-error-code' with value "404".</p>	<p>The referenced destination or fragment is not found in the destination service for the given tenant.</p>
<p>The call to the Kubernetes service referencing your destination returns response with status code 400, response header 'x-error-message' with value "The '<headerKey>' header should be passed only once. Check your request parameters." response header 'x-error-origin' with value "Transparent Proxy",</p>	<p>The provided header should be passed only once.</p>
<p>Applying helm configuration fails with "When .Values.config.security.communication.internal.encryptionEnabled is true, .Values.config.security.communication.internal.certManager.issuerRef.name cannot be empty".</p>	<p>Set value to .Values.config.security.communication.internal.certManager.issuerRef.name when .Values.config.security.communication.internal.encryptionEnabled is true.</p>
<p>Applying helm configuration fails with ".Values.config.security.communication.internal.certManager.issuerRef.namespace and .Values.config.security.communication.internal.certManager.issuerRef.kind are both passed. Use .Values.config.security.communication.internal.certManager.issuerRef.kind when your issuer is of type cert-manager.io and .Values.config.security.communication.internal.certManager.issuerRef.namespace when your issuer is of type cert.gardener.cloud".</p>	<p>Pass either .Values.config.security.communication.internal.certManager.issuerRef.kind or .Values.config.security.communication.internal.certManager.issuerRef.namespace, not both, when .Values.config.security.communication.internal.encryptionEnabled is true.</p>
<p>Applying helm configuration fails with "Neither .Values.config.security.communication.internal.certManager.issuerRef.namespace or .Values.config.security.communication.internal.certManager.issuerRef.kind is passed. Use .Values.config.security.communication.internal.certManager.issuerRef.kind when your issuer is of type cert-manager.io and .Values.config.security.communication.internal.certManager.issuerRef.namespace when your issuer is of type cert.gardener.cloud".</p>	<p>One of .Values.config.security.communication.internal.certManager.issuerRef.kind and .Values.config.security.communication.internal.certManager.issuerRef.namespace is required when .Values.config.security.communication.internal.encryptionEnabled is true.</p>
<p>Applying helm configuration fails with "Issuer of version cert.gardener.cloud/v1alpha1 with name 'test' in namespace 'test' not found".</p>	<p>Make sure that Issuer of version cert.gardener.cloud/v1alpha1 with name 'test' in namespace 'test' exists</p>
<p>Applying helm configuration fails with "Issuer of version cert-manager.io/v1 with name 'test' in namespace 'test' not found".</p>	<p>Make sure that Issuer of version cert-manager.io/v1 with name 'test' in namespace 'test' exists</p>

Issue	Solution
Applying helm configuration fails with "ClusterIssuer of version cert-manager.io/v1 with name 'test' not found".	Make sure that ClusterIssuer of version cert-manager.io/v1 with name 'test' exists
The destinations.destination.connectivity.api.sap custom resource referencing your destination with the name "testdest" has the name "testservice". The last condition of type "Available" of the destinations.destination.connectivity.api.sap custom resource has the status "False" with the reason "ConfigurationError-K8sResourceWithNameAlreadyPresent".	There is a Kubernetes resource with the name of the Destination custom resource present in the same namespace. Rename one of the resources to remove the collision.
Applying helm configuration fails with "Istio not installed on the cluster. Revise your Istio integration configuration."	The configuration for integration in Istio service mesh is provided but Istio is not installed on the cluster. Either install Istio on the cluster or remove the configuration for integration in Istio service mesh.
Applying helm configuration fails with "An error occurred because certain necessary configuration settings are missing. Ensure that either 'config.security.communication.internal.encryptionEnabled' or 'config.integration.serviceMesh.istio.istio-integration' is provided."	It is mandatory to integrate either with Istio or cert-manager. This is done by configuring at least one of the two properties:: config.integration.serviceMesh.istio.istio-integration and config.security.communication.internal.encryptionEnabled. config.integration.serviceMesh.istio.istio-integration makes config.security.communication.internal.encryptionEnabled optional. If "encryptionEnabled" is set to true and config.integration.serviceMesh.istio.istio-integration is missing then config.security.communication.internalCommunication.certManager.issuerRef.name and config.security.communication.internalCommunication.certManager.issuerRef.kind have to be provided.
Applying helm configuration fails with 'Connectivity Proxy tenant mode "shared" is not compatible with Transparent Proxy tenant mode "dedicated". Both must be in the same mode.'	Make sure that the transparent proxy and the referenced connectivity proxy are in the same tenant mode.
Applying helm configuration fails with 'Connectivity Proxy tenant mode "dedicated" is not compatible with Transparent Proxy tenant mode "shared". Both must be in the same mode.'	Make sure that the transparent proxy and the referenced connectivity proxy are in the same tenant mode.
Applying helm configuration fails with "Connectivity Proxy multi region ConfigMap with name <connectivityProxyMultiRegionConfigMapName> in namespace <connectivityProxyMultiRegionConfigMapNamespace> does not exist!".	The connectivity proxy multi region config map does not exist.
The call to the Kubernetes service referencing your destination returns response with status code 502, response header 'x-error-message' with value "Region with ID <regionId> is not found in the Connectivity Proxy configuration.", response header 'x-error-origin' with value " Transparent Proxy".	The region id <regionId> passed as SAP-Connectivity-Region-Configuration-Id header or statically configured for the given destination service instance in the transparent proxy configurations is not found in the connectivity proxy multi region configurations. Make sure that the used region id is existing in the connectivity proxy multi region configurations.

Issue	Solution
The call to the Kubernetes service referencing your destination returns response with status code 400, response header 'x-error-message' with value "Connectivity Proxy region ID is not specified. Either pass it as a 'SAP-Connectivity-Region-Configuration-Id' or set it in the Transparent Proxy configuration.", response header 'x-error-origin' with value "Transparent Proxy".	You should either pass SAP-Connectivity-Region-Configuration-Id header or statically configure the region for the given destination service instance in the transparent proxy configurations.
The call to the Kubernetes service referencing your destination returns response with status code 502, response header 'x-error-message' with value "Invalid Connectivity proxy instance configuration. Contact the local Kubernetes cluster administrator to inspect the Connectivity Proxy deployment configuration.", response header 'x-error-origin' with value "Transparent Proxy".	The connectivity proxy multi region config map does not exist.
The call to the Kubernetes service referencing your destination returns response with status code 502, response header 'x-error-message' with value "The referenced Connectivity Proxy instance configuration for region named <regionName> is not found or is invalid."	The secret referenced by the connectivity proxy region configuration named <regionName> does not exist or is invalid.

1.4.8 Recommended Actions

To resolve issues with the transparent proxy for Kubernetes, follow the recommendations below.

Pre-Intervention Steps

Before doing any restarts or modifications, it is important that you collect all relevant information.

1. Check the status of the transparent proxy pods and collect the outcome. Example via `kubectl`:

1. Get the status of the transparent proxy manager executions:

```
kubectl get pods -l transparent-proxy.connectivity.api.sap/
component=manager -n <installation-namespace>
```

2. Get the status of the transparent HTTP proxy pods:

```
kubectl get pods -l transparent-proxy.connectivity.api.sap/component=http
-n <installation-namespace>
```

3. Get the status of the transparent TCP proxy pods:

```
kubectl get pods -l transparent-proxy.connectivity.api.sap/component=tcp -
n <installation-namespace>
```

2. Perform basic availability monitoring from within the cluster as described in [Monitoring \[page 840\]](#), and collect the outcome.
3. Collect the logs of all transparent proxy components (see [Troubleshooting \[page 841\]](#)).
4. Collect the status of all `destinations.destination.connectivity.api.sap` custom resources (see [Troubleshooting \[page 841\]](#)).
5. Proceed with [Recovery Attempt \[page 855\]](#) below.

Recovery Attempt

The exact action to take here depends on the check result in step 2 of section [Pre-Intervention Steps \[page 854\]](#). Choose one of the two options below, according to the outcome of your checks.

- Option 1: [Check Succeeds \[page 855\]](#)
- Option 2: [Check Fails \[page 856\]](#)

Check Succeeds

This indicates that the transparent proxy is currently considered operational. However, it might still have trouble when used for real scenarios (depending on how you detect the outage).

1. Check if the outage is still ongoing:
 1. If no, issue is resolved, proceed with [Request Root Cause Analysis\(RCA\) \[page 856\]](#).
 2. If yes, issue is not resolved, proceed with the next step.
2. Wait for the next transparent proxy manager completion.
3. Check if the outage is still ongoing:
 1. If no, issue is resolved, proceed with [Request Root Cause Analysis\(RCA\) \[page 856\]](#).
 2. If yes, issue is not resolved, proceed with the next step.
4. Restart the transparent HTTP proxy and transparent TCP proxy instances. Example via `kubectl`:

```
k delete po -l transparent-proxy.connectivity.api.sap/component=http -n <installation-namespace>
```

```
k delete po -l transparent-proxy.connectivity.api.sap/component=tcp -n <installation-namespace>
```

5. Collect the logs from the transparent proxy components after the restart completes.
6. Check if outage is still ongoing:
 1. If no, issue is resolved, proceed with [Request Root Cause Analysis\(RCA\) \[page 856\]](#).
 2. If yes, issue is not resolved, proceed with [Request Help from SAP \[page 856\]](#).

Check Fails

This indicates that the transparent proxy itself is indeed having issues. Please perform the following steps:

1. Wait for the next transparent proxy manager completion.
2. Check if the outage is still ongoing:
 1. If no, issue is resolved, proceed with [Request Root Cause Analysis\(RCA\) \[page 856\]](#).
 2. If yes, issue is not resolved, proceed with the next step.
3. Restart the transparent HTTP proxy and transparent TCP proxy instances. Example via `kubectl`:

```
k delete po -l transparent-proxy.connectivity.api.sap/component=http -n <installation-namespace>
```

```
k delete po -l transparent-proxy.connectivity.api.sap/component=tcp -n <installation-namespace>
```

4. Collect logs from the transparent proxy after the restart completes.
5. Check if outage is still ongoing:
 1. If no, issue is resolved, proceed with [Request Root Cause Analysis\(RCA\) \[page 856\]](#).
 2. If yes, issue is not resolved, proceed with [Request Help from SAP \[page 856\]](#).

Request Help from SAP

If you cannot resolve the issue and require help from SAP, follow this procedure:

1. Open an incident on the support component BC-CP-CON-K8S-TP for the transparent proxy (with the appropriate priority and impact stated).
2. Provide all the collected information in the incident, including all the logs, timestamps of the events that occurred, a summary of the taken actions, the version of the transparent proxy you are using, and so on.
3. Engage your SAP contacts to help with this.
4. Continue working in parallel to identify as much information as possible or to find a temporary measure to mitigate the outage.

Request Root Cause Analysis (RCA)

Once the issue is resolved, the next step is figuring out what exactly caused the issue and if there is something that can be done to prevent it from happening in the future.

Follow this procedure for requesting RCA for the issue you experienced:

1. Open an incident on the support component for the transparent proxy on the support component BC-CP-CON-K8S-TP.
2. Provide all the collected information in the incident, including all the logs, timestamps of the events that occurred, a summary of the taken actions, the version of the transparent proxy you are using, and so on.

The incident will be handled according to the SAP incident SLAs.

1.4.9 Resilience

Improve resilience of the transparent proxy for Kubernetes.

Even though the transparent proxy is not a cloud service, it offers options to improve resilience and robustness of the whole system benefiting from the transparent proxy functionality.

The software architecture of the transparent proxy in terms of HTTP and TCP proxies is loosely decoupled, meaning that if there are failures in one of them it would not affect the other. Thus they are independent of each other and errors regarding for example the transparent HTTP proxy should not affect the work of the transparent TCP proxy instances.

To achieve additional resilience, see [High Availability \[page 857\]](#).

1.4.9.1 High Availability

Run the transparent proxy for Kubernetes in a high-availability setup.

Overview

The transparent proxy can work in an active-active high-availability setup. In this setup, there are at least two transparent proxy instances, both running actively and simultaneously.

The main purpose of an active-active deployment is to provide high availability and allow zero-downtime maintenance as well as horizontal-scaling capabilities.

ⓘ Note

The load balancing strategy for distributing traffic to the connectivity proxy pods depends on the kube-proxy mode. The strategies used by the different kube-proxy modes are described in the [Kubernetes documentation](#). This configuration is done on cluster level.

Scaling

If you need to scale your transparent proxy, you can do so by updating your Helm chart. Follow these steps:

1. For horizontal scaling of the TCP and HTTP proxy, update the `values.yaml`, and more concretely `deployment.replicas.tcp` for HTTP and `deployment.replicas.tcp` for TCP, as described in the [Configuration \[page 823\]](#) section.
2. For vertical scaling, update the respective `deployment.resources.limits` and `deployment.resources.requests` properties for *CPU* and *Memory* described in the [Configuration \[page 823\]](#) section.
3. To enable autoscaling of the transparent proxy, update the respective `deployment.autoscaling.*` properties.

ⓘ Note

Horizontal and vertical autoscaling cannot be activated simultaneously. You have to choose one or the other.

Multi-AZ

If the Kubernetes cluster is configured with two or more nodes running in different availability zones (AZ), the transparent proxy automatically deploys and runs in the AZs for the Kubernetes cluster that matches the nodes.

1.4.10 Verification and Testing

Check the transparent proxy for Kubernetes after installation.

Once you have installed the transparent proxy in your cluster, you can perform the following steps to verify it is running successfully.

ⓘ Note

You may have to wait for a few seconds before all the components are started and can be consumed.

To check the status of the transparent proxy's components, execute the following, replacing the namespace placeholder:

```
kubectl run perform-hc --image=radial/busyboxplus:curl -it --rm --restart=Never -- curl -w "\n" 'sap-transp-proxy-int-healthcheck.{installation-namespace}/status'
```

or the following commands when your transparent proxy is started within *Istio Service Mesh*:

```
kubectl run perform-hc --image=radial/busyboxplus:curl -- labels="sidecar.istio.io/inject=true" -it --rm
```

```
curl -w '\n' sap-transp-proxy-int-healthcheck.{installation-namespace}/status
exit
```

and observe the result.

Example:

```
{
  "Status": "ok", // possible values: "ok|warning|critical"
  "Components": {
    "sap-transp-proxy-http": {
      "Status": "ok", // possible values: "ok|warning|critical"
      "Message": "" // optional if status is not ok.
    },
    "sap-transp-proxy-tcp": {
      "Status": "ok", // possible values: "ok|warning|critical"
      "Message": "", // optional if status is not ok.
      "AffectedDeployments": []// optional if status is not ok. Lists all
sap-transp-proxy-tcp deployments that have unready pods.
    },
    "sap-transp-proxy-manager": { "Status": "ok", // possible values: "ok|
warning|critical"
      "Message": "" // optional if status is not ok.
    }
  }
}
```

To check if all destination custom resources are configured properly, execute:

```
kubectrl run perform-hc --image=curlimages/curl -it --rm --restart=Never
-- curl -w "\n" 'sap-transp-proxy-int-healthcheck.{installation-namespace}/
destinationCRs'
```

and observe the result.

Example:

```
{
  "Status": "ok",
  "CustomResources": [{
    "Name": "httpexample",
    "DestinationRef": {
      "name": "httpdestination"
    },
    "CreationTimestamp": "2022-08-26T11:05:30Z",
    "Available": "True",
    "Message": "Technical connectivity is configured. Kubernetes service
with name httpexample is created.",
    "Reason": "ConfigurationSuccessful"
  }, {
    "Name": "tcpexample",
    "DestinationRef": {
      "name": "tcpdestination"
    },
    "CreationTimestamp": "2022-08-26T11:05:34Z",
    "Available": "True",
    "Message": "Technical connectivity is configured. Kubernetes service
with name tcpexample is created.",
    "Reason": "ConfigurationSuccessful"
  }
]
```

```
} ]  
}
```

If you encounter problems with any of the above steps, see [Troubleshooting \[page 841\]](#) and [Recommended Actions \[page 854\]](#) for further investigation.

Now you can proceed with consuming the transparent proxy from your Kubernetes applications.

1.4.11 Transparent Proxy in the Kyma Environment

Use the transparent proxy in the Kyma environment.

This documentation serves as a reference for the transparent proxy enablement in the Kyma environment and the respective transparent proxy operator, integrated with [Kyma's Lifecycle Manager](#) .

The transparent proxy operator continuously observes the state of the system and the desired state defined by the transparent proxy custom resource. It then makes necessary adjustments to the system (like creating, updating, or deleting resources) to achieve the desired state, and regularly monitors the health of the transparent proxy, ensuring it runs optimally according to the configurations defined in the custom resource.

Module Enablement

For module enablement, refer to [Enable and Disable a Kyma module](#).

With the enablement of the transparent proxy module, a default transparent proxy custom resource is created in the **sap-transp-proxy-system** namespace.

⚠ Caution

The **sap-transp-proxy-system** namespace is intended for the monitoring of the transparent proxy resources, do not use it for deploying workloads there.

The default transparent proxy custom resource is defined like this:

```
apiVersion: operator.kyma-project.io/v1alpha1  
kind: TransparentProxy  
metadata:  
  name: transparent-proxy  
  namespace: sap-transp-proxy-system  
spec:  
  config:  
    security:  
      communication:  
        internal:  
          encryptionEnabled: true  
  integration:  
    serviceMesh:  
      istio:
```

```
istio-injection: enabled
```

You can update it to meet your requirements. To do this, either use `kubectl edit` or create a YAML file containing the transparent proxy custom resource and edit the spec section with values according to the [Configuration Guide \[page 822\]](#).

Sample Code

```
kubectl apply -f <fileName> -n sap-transp-proxy-system
```

If you want to create a second instance of the transparent proxy, you can create a transparent proxy custom resource in another namespace.

Dependencies

Service Mesh Configuration

The transparent proxy supports integration with the Istio service mesh (for more information, see [Configuration Guide \[page 822\]](#)).

If the transparent proxy is configured to integrate in the *Istio mesh* and Istio is present on the cluster, the integration with the *certificate manager* becomes optional, otherwise it is mandatory.

If `encryptionEnabled` is set to "false", but there is integration in the *Istio service mesh*, the transparent proxy custom resource will assume the state "Ready" with message *"installation is ready. Although encryptionEnabled is set to false, the traffic will be encrypted by Istio."*. You can use both, integration with *Istio service mesh* and with the *certificate manager*.

Encryption between Micro Components

If you don't integrate the transparent proxy with a service mesh, you must encrypt the traffic between the micro components. To do that, set `encryptionEnabled` to `true`. This property configures encryption between the transparent proxy micro components internally. Additionally, you should provide the necessary *certificate manager* configuration to make sure encryption works as expected.

For more information, see [Configuration Guide \[page 822\]](#).

Destination Service Instance

The transparent proxy will load all resources of api version `services.cloud.sap.com/v1` and kind `ServiceInstance` having `spec.serviceOfferingName: destination`, created in namespace `sap-transp-proxy-system`, as destination service instances. In addition, you can configure more destination service instances directly in the transparent proxy custom resource.

For more information, see the [Configuration Guide \[page 822\]](#).

If no resources of api version `services.cloud.sap.com/v1` and kind `ServiceInstance` exist in namespace `sap-transp-proxy-system`, and no destination service instances are directly specified in the transparent proxy

custom resource, a default, empty destination service instance with name `sap-transp-proxy-default` will be created in namespace `sap-transp-proxy-system` and loaded as destination service instance in the transparent proxy custom resource.

Connectivity Proxy

The transparent proxy will automatically connect with the connectivity proxy if the Kyma module for the connectivity proxy is enabled. If you choose to manually link to a connectivity proxy, the automatic connection will not override your configuration.

Troubleshooting

In case of issues, please refer to [Troubleshooting in Kyma \[page 862\]](#).

1.4.11.1 Troubleshooting in Kyma

Find solutions for issues with the transparent proxy in Kyma.

Common Issues and Solutions

For custom resources having a *Warning* state, refer to this table to find a solution for specific issues.

Check the [Configuration Guide \[page 822\]](#) after identifying your misconfiguration in the transparent proxy custom resource conditions.

Custom Resource Condition Message	Solution
<p>An error occurred because certain necessary configuration settings are missing. Make sure either <code>config.security.communication.internal.encryptedEnabled</code> or <code>config.integration.serviceMesh.istio.istio-integration</code> is provided.</p>	<p>Integration either with <i>Istio</i> or <i>cert-manager</i> is mandatory.</p> <p>This is done by configuring at least one of the two properties: <code>config.integration.serviceMesh.istio.istio-integration</code> or <code>config.security.communication.internal.encryptedEnabled</code>.</p> <ul style="list-style-type: none"> <code>config.integration.serviceMesh.istio.istio-integration</code> makes <code>config.security.communication.internal.encryptedEnabled</code> optional. If "encryptedEnabled" is set to true and <code>config.integration.serviceMesh.istio.istio-integration</code> is missing, you must provide <code>config.security.communication.internalCommunication.certManager.issuerRef.name</code> and <code>config.security.communication.internalCommunication.certManager.issuerRef.kind</code>.
<p>Custom resource with name "<name>" already exists in this namespace.</p>	<p>There could be only 1 transparent proxy custom resource in a single namespace. Delete the unnecessary custom resource or move it to another namespace.</p>
<p>When <code>config.security.communication.internal.encryptedEnabled</code> is true, <code>config.security.communication.internal.certManager.issuerRef.name</code> cannot be empty.</p>	<p>If <code>config.security.communication.internalCommunication.encryptedEnabled</code> is set to true, then <code>config.security.communication.internal.certManager.issuerRef.name</code> cannot be empty.</p>
<p><code>config.security.communication.internal.certManager.issuerRef.namespace</code> and <code>config.security.communication.internal.certManager.issuerRef.kind</code> are both passed.</p> <p>Use <code>config.security.communication.internal.certManager.issuerRef.kind</code> when your issuer is of type <code>cert-manager.io</code> and <code>config.security.communication.internal.certManager.issuerRef.namespace</code> when your issuer is of type <code>cert.gardener.cloud</code>.</p>	<p>If <code>config.security.communication.internalCommunication.encryptedEnabled</code> is set to true, you must pass either <code>config.security.communication.internal.certManager.issuerRef.kind</code> or <code>config.security.communication.internal.certManager.issuerRef.namespace</code>, not both.</p>

Custom Resource Condition Message	Solution
<p>Neither</p> <pre>config.security.communication.internal.certManager.issuerRef.namespace</pre> <p>nor</p> <pre>config.security.communication.internal.certManager.issuerRef.kind</pre> <p>is passed.</p> <p>Use</p> <pre>config.security.communication.internal.certManager.issuerRef.kind</pre> <p>when your issuer is of type cert-manager.io and</p> <pre>config.security.communication.internal.certManager.issuerRef.namespace</pre> <p>when your issuer is of type <code>cert.gardener.cloud</code>.</p>	<p>If</p> <pre>config.security.communication.internalCommunication.encryptionEnabled</pre> <p>is set to true, you must pass either</p> <pre>config.security.communication.internal.certManager.issuerRef.kind</pre> <p>or</p> <pre>config.security.communication.internal.certManager.issuerRef.namespace</pre> <p>, both cannot be empty.</p>
"certificates.cert-manager.io" custom resource definition not found.	<p>If</p> <pre>config.security.communication.internalCommunication.encryptionEnabled</pre> <p>is set to true, cert-manager must be installed in the cluster.</p>
"certificates.cert.gardener.cloud" custom resource definition not found,	<p>If</p> <pre>config.security.communication.internalCommunication.encryptionEnabled</pre> <p>is set to true and</p> <pre>config.security.communication.internalCommunication.certManager.issuerRef.namespace</pre> <p>is not empty, then cert-manager-gardener must be installed in the cluster.</p>
<pre>config.security.communication.internalCommunication.certManager.issuerRef</pre> <p>properties: [kind and name] should be provided.</p>	<p>If</p> <pre>config.security.communication.internalCommunication.encryptionEnabled</pre> <p>is set to true, a valid reference to an existing <code>Issuer</code> or <code>ClusterIssuer</code> must be provided to secure the internal transparent proxy communication by mTLS.</p>
Resource with api version <code>cert-manager.io/v1</code> and kind " <code><Issuer ClusterIssuer></code> " with name " <code><name></code> " not found.	The referenced <code>Issuer</code> or <code>ClusterIssuer</code> is not found in the cluster. If it is of type <code>Issuer</code> , check the namespace and name. For <code>ClusterIssuer</code> , check the name.
Resource with api version <code>cert.gardener.cloud/v1alpha1</code> and kind " <code>Issuer</code> " in namespace " <code><namespace></code> " with name " <code><name></code> " not found.	The referenced <code>Issuer</code> with api version <code>cert.gardener.cloud/v1alpha1</code> is not found. Check the specified namespace to see if the resource exists.
Provide at least one <code>Destination</code> service instance in <code>config.integration.destinationServiceIntegration.instances</code> , or create and bind an instance in the <code>sap-transp-proxy-system</code> namespace using the <i>SAP BTP Service Operator</i> .	The transparent proxy should have at least one <code>Destination</code> service instance configured in <code>config.integration.destinationServiceIntegration.instances</code> , or resources of api version " <code>services.cloud.sap.com/v1</code> " and kind " <code>ServiceInstance</code> " should exist in the " <code>sap-transp-proxy-system</code> " namespace.
Destination service instance name should be provided.	Some of the entries in <code>config.integration.destinationService.instances</code> does not have name specified.

Custom Resource Condition Message	Solution
<code>secretName</code> should be provided for Destination service instance: " <code><name></code> ".	The given destination service instance should have valid reference to a secret holding the service credentials for consuming the Destination service.
<code>secret</code> with name " <code><name></code> " not found in namespace: " <code><namespace></code> ".	The provided secret cannot be found in the referenced namespace. Examples: <ul style="list-style-type: none"> The referenced secret holding the service credentials for the Destination service is not found for the referenced destination service instance. Check the <code>secretName</code> and <code>secretNamespace</code> properties provided in the <code>serviceCredentials</code> section for the given instance in <code>config.integration.destinationService.instances</code>. The referenced secret holding the service credentials for the connectivity proxy is not found. Check the <code>secretName</code> and <code>secretNamespace</code> properties provided in the <code>serviceCredentials</code> section for the given instance in <code>config.integration.connectivityProxy.serviceCredentials</code>.
<code>secretName</code> should be provided for connectivity proxy service instance in shared mode.	<code>config.integration.connectivityProxy.serviceCredentials.secretName</code> should be provided when <code>config.integration.connectivityProxy.serviceName</code> is not blank, and the connectivity proxy referenced by <code>config.integration.connectivityProxy.serviceName</code> is in shared mode.
<code>configMap</code> " <code>connectivity-proxy</code> " not found in namespace " <code><namespace></code> ".	The given config map is not present in the same namespace as the service defined in <code>config.integration.connectivityProxy.serviceName</code> . If the service name contains a namespace, the config map should be in that namespace otherwise it should be in the same namespace as the transparent proxy custom resource.
Key " <code>connectivity-proxy-config.yml</code> " not found in <code>configMap</code> 's Data from <code>configMap</code> : " <code>connectivity-proxy</code> " in namespace " <code><namespace></code> ".	The referenced config map does not contain the expected " <code>connectivity-proxy-config.yml</code> " Data key.
Connectivity proxy tenant mode " <code><tenant_mode></code> " is not compatible with transparent proxy tenant mode " <code><tenant_mode></code> ".	The tenant mode defined in the connectivity proxy's <code>configmap</code> is different from the one defined in <code>config.tenantMode</code> of the transparent proxy custom resource. The tenant modes of the connectivity proxy and transparent proxy must be equal (shared & shared or dedicated & dedicated).

Custom Resource Condition Message	Solution
Istio not installed on the cluster. Revise your Istio integration configuration.	The configuration for integration in Istio service mesh is provided, but Istio is not installed on the cluster. Either install Istio on the cluster or remove the configuration for integration in Istio service mesh.

1.4.12 Dynamic Lookup of Destinations

Create a single custom resource to look up one or more destinations dynamically with the transparent proxy for Kubernetes.

The transparent proxy lets you perform a dynamic lookup of one or multiple destinations from a destination service instance and its tenants. To consume a destination in that way, you only have to create a single custom resource.

Example: Destination Custom Resource

```

apiVersion: destination.connectivity.api.sap/v1
kind: Destination
metadata:
  name: dynamic-destination
spec:
  destinationRef:
    name: "*"
  destinationServiceInstanceName: dest-service-instance //not mandatory

```

In the example above, a destination custom resource has `spec.destinationRef.name = "*"` , which indicates that this destination would accept dynamic lookup and only a single Kubernetes service will be created with name `dynamic-destination` which works as an entry point for all destinations from `destinationServiceInstanceName` with `name: dest-service-instance`.

The transparent proxy identifies the destination for which a request should be configured and dispatched by obtaining the destination name (destination identifier) from a custom header called `X-Destination-Name`.

Note

This header is specifically intended to identify a destination within a Destination service instance and its tenants.

The destination can also be combined by optionally providing a custom header called `X-Fragment-Name`.

Example Destination:

```

{
  "Name": "example-dest-client-cert",
  "Type": "HTTP",
  "ProxyType": "Internet",
  "URL": "https://myapp.com",
  "Authentication": "ClientCertificateAuthentication",
  "KeyStorePassword": "Abcd1234",
  "KeyStoreLocation": "cert.jks"
}

```

```
}
```

Example Fragment:

```
{  
  "FragmentName": "example-fragment",  
  "URL": "https://myotherapp.com"  
}
```

If you want to call the `example-dest-client-cert` destination through the Kubernetes service named `dynamic-destination`, you can execute the command below:

```
curl dynamic-destination.<transparent-proxy-namespace> -H "X-Destination-Name:  
example-dest-client-cert"
```

If you want to call `example-dest-client-cert` destination, enriching it with the properties from fragment `example-fragment` through the Kubernetes service named `dynamic-destination`, you can execute the command below:

```
curl dynamic-destination.<transparent-proxy-namespace> -H "X-Destination-Name:  
example-dest-client-cert" -H "X-Fragment-Name: example-fragment"
```

1.5 Connectivity via Reverse Proxy

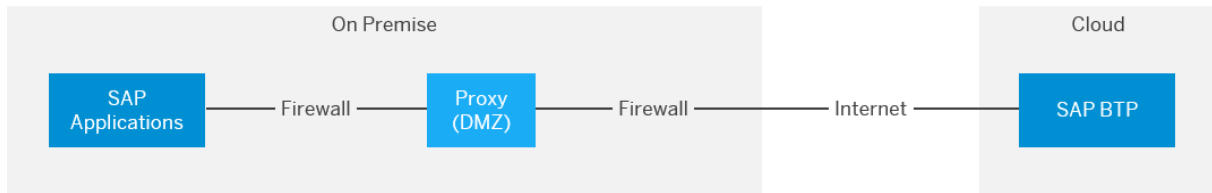
The text discusses the use of a reverse proxy as an alternative approach to connect on-premise services to SAP BTP. While it allows for reuse of existing network infrastructure, it exposes services to potential attacks and requires significant involvement from your IT department. The Cloud Connector is recommended as a more secure and efficient solution, providing TLS tunneling and fine-grained access control.

An alternative approach compared to the TLS tunnel solution that is provided by the Cloud Connector is to expose on-premise services and applications via a reverse proxy to the Internet. This method typically uses a reverse proxy setup in a customer's "demilitarized zone" (DMZ) subnetwork. The reverse proxy setup does the following:

- Acts as a mediator between SAP BTP and the on-premise services
- Provides the services of an Application Delivery Controller (ADC) to, for example, encrypt, filter, route, or check inbound traffic

The figure below shows the minimal overall network topology of this approach.

On-premise services that are accessible via a reverse proxy are callable from SAP BTP like other HTTP services available on the Internet. When you use destinations to call those services, make sure the configuration of the `ProxyType` parameter is set to **Internet**.



Advantages

Depending on your scenario, you may benefit from the reverse proxy:

- Network infrastructure (such as a reverse proxy and ADC services): since it already exists in your network landscape, you can reuse it to connect to SAP BTP. There's no need to set up and operate new components on your (customer) side.
- A reverse proxy is independent of the cloud solution you are using.
- It acts as single entry point to your corporate network.

Disadvantages

- The reverse proxy approach leaves exposed services generally accessible via the Internet. This makes them vulnerable to attacks from anywhere in the world. In particular, **Denial-of-Service attacks** are possible and difficult to protect against. To prevent attacks of this type and others, you must implement the highest security in the DMZ and reverse proxy. For the productive deployment of a hybrid cloud/on-premise application, this approach usually requires intense involvement of the customer's IT department and a longer period of implementation.
- If the reverse proxy allows filtering, or restricts accepted source IP addresses, you can set only one IP address to be used for all SAP BTP outbound communications. A reverse proxy does not exclusively restrict the access to cloud applications belonging to a customer, although it does filter any callers that are not running on the cloud. Basically, any application running on the cloud would pass this filter.
- The SAP-proprietary RFC protocol is supported only if WebSocket RFC can be used for communication with the ABAP system. WebSocket RFC is available as of S/4HANA release 1909. A cloud application cannot call older on-premise ABAP systems directly without using application proxies on top of ABAP in between.
- No easy support of principal propagation authentication, which lets you forward the cloud user identity to on-premise systems.
- You cannot implement projects close to your line of business (LoB).

📌 Note

Using the Cloud Connector mitigates all of these issues. As it establishes the TLS tunnel to SAP BTP using a reverse invoke approach, there is no need to configure the DMZ or external firewall of a customer network for inbound traffic. Attacks from the Internet are not possible. With its simple setup and fine-grained access control of exposed systems and resources, the Cloud Connector allows a high level of security and fast


productive implementation of hybrid applications. It also supports multiple application protocols, such as HTTP and RFC.

1.6 Connectivity Support

Support information for SAP BTP Connectivity and the Cloud Connector.

Troubleshooting

Locate the problem or error you have encountered and follow the recommended steps:

- [Frequently Asked Questions \[page 719\]](#) (Cloud Connector)
- [Administration \[page 628\]](#) (Cloud Connector)
- [Cloud Connectivity: Guided Answers](#) 
- [Getting Support](#) (SAP Support Portal, SAP BTP community)

SAP Support Information

If you cannot find a solution to your issue, collect and provide the following specific, issue-relevant information to SAP Support:

- The Java EE code that throws an error (if any)
- A screenshot of the error message for the failed operation or the error message from the `HttpResponse` body
- Access credentials for your cloud or on-premise location

You can submit this information by creating a customer ticket in the SAP CSS system using the following components:

Component	Purpose
Connectivity Service	
BC-CP-CON	For <i>cloud-side</i> issues with cloud to on-premise connectivity, where: <ul style="list-style-type: none">• The environment is unknown or• The issue is not related to a specific environment
BC-CP-CON-CF	For <i>cloud-side</i> issues with cloud to on-premise connectivity in the SAP BTP Cloud Foundry environment .

Component	Purpose
BC-CP-CON-S4HC	For <i>cloud-side</i> issues with cloud to on-premise connectivity in an S/4HANA Cloud system.
BC-CP-CON-K8S-PROXY	For cloud-side issues with cloud to on-premise connectivity in a Kubernetes cluster (or Kubernetes-based product), using the connectivity proxy software component.
BC-CP-CON-ABAP	For <i>cloud-side</i> issues with cloud to on-premise connectivity in the SAP BTP ABAP environment .
BC-NEO-CON	For <i>cloud-side</i> issues with cloud to on-premise connectivity in the SAP BTP Neo environment .
Destinations	
BC-CP-DEST	For issues with destination configurations , where: <ul style="list-style-type: none"> • The environment is unknown or • The issue is not related to a specific environment
BC-CP-DEST-CF	For general issues with the Destination service in the SAP BTP Cloud Foundry environment , like: <ul style="list-style-type: none"> • REST API • Instance creation, etc.
BC-CP-DEST-CF-CLIBS	For client library issues with the Destination service in the SAP BTP Cloud Foundry environment .
BC-CP-DEST-CF-TOOLS	For issues with the management of destination configurations via tools like the SAP BTP cockpit (Cloud Foundry environment).
BC-CP-DEST-NEO	For issues with destination configurations or: <ul style="list-style-type: none"> • Management tools • Client libraries, etc. related to destinations in the SAP BTP Neo environment .
Cloud Connector	
BC-MID-SCC	For connectivity issues related to installing and configuring the Cloud Connector , configuring tunnels, connections, and so on.

If you experience a more serious issue that cannot be resolved using only traces and logs, SAP Support may request access to the Cloud Connector. Follow the instructions in these SAP notes:

- To provide access to the Administration UI via a browser, see [592085](#).
- To provide SSH access to the operating system of the Linux machine on which the connector is installed, see [1275351](#).

Related Information

[Release and Maintenance Strategy \[page 871\]](#)

1.6.1 Release and Maintenance Strategy

Find information about SAP BTP Connectivity releases, versioning and upgrades.

Release Cycles

Updates of the Connectivity service are published as required, within the regular, bi-weekly SAP BTP release cycle.

New releases of the Cloud Connector are published when new features or important bug fixes are delivered, available on the [Cloud Tools](#) page.

Cloud Connector Versions

Cloud Connector versions follow the `<major>.<minor>.<micro>` versioning schema. The Cloud Connector stays fully compatible within a major version. Within a minor version, the Cloud Connector will stay with the same feature set. Higher minor versions usually support additional features compared to lower minor versions. Micro versions generally consist of patches to a `<major>.<minor>` version to deliver bug fixes.

For each supported major version of the Cloud Connector, only one `<major>.<minor>.<micro>` version will be provided and supported on the Cloud Tools page. This means that users must upgrade their existing Cloud Connectors to get a patch for a bug or to make use of new features.

For detailed support strategy information, check SAP note [3302250](#) .

Cloud Connector Upgrade

New versions of the Cloud Connector are announced in the [Release Notes](#) of SAP BTP. We recommend that Cloud Connector administrators regularly check the release notes for Cloud Connector updates. New versions of the Cloud Connector can be applied by using the Cloud Connector upgrade capabilities. For more information, see [Upgrade \[page 715\]](#).

Note

We recommend that you first apply upgrades in a test landscape to validate that the running applications are working.



There are no manual user actions required in the Cloud Connector when the SAP BTP is updated.

Important Disclaimers and Legal Information

Hyperlinks

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

About the icons:

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

Videos Hosted on External Platforms

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

Beta and Other Experimental Features

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

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

Example Code

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

Bias-Free Language

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

© 2024 SAP SE or an SAP affiliate company. All rights reserved.

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

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

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

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

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