



# SAP BTP Connectivity

Generated on: 2023-05-19 11:35:45 GMT+0000

SAP BTP Connectivity | Cloud

PUBLIC

Original content: [https://help.sap.com/docs/CP\\_CONNECTIVITY/cca91383641e40ffbe03bdc78f00f681?locale=en-US&state=PRODUCTION&version=Cloud](https://help.sap.com/docs/CP_CONNECTIVITY/cca91383641e40ffbe03bdc78f00f681?locale=en-US&state=PRODUCTION&version=Cloud)

## Warning

This document has been generated from the SAP Help Portal and is an incomplete version of the official SAP product documentation. The information included in custom documentation may not reflect the arrangement of topics in the SAP Help Portal, and may be missing important aspects and/or correlations to other topics. For this reason, it is not for productive use.

For more information, please visit the <https://help.sap.com/docs/disclaimer>.

# Connectivity

SAP BTP Connectivity: overview, features, restrictions.

## **i 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

Please note that image maps are not interactive in PDF output.

### In this Guide

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

Cloud Foundry Environment

Cloud Connector

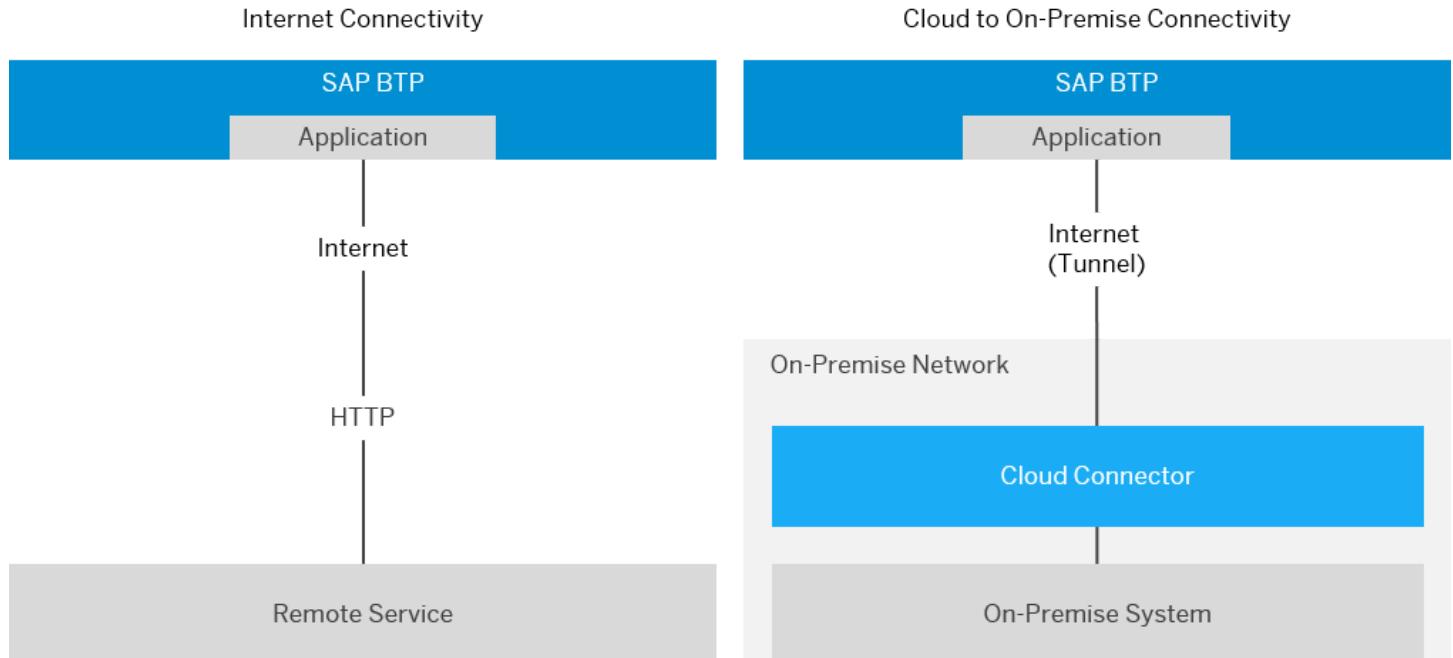
Support

Please note that image maps are not interactive in PDF output.

## 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](#) (on-premise connections) or the respective API endpoints of your SAP BTP [region](#) (Internet connections).

Back to [Content](#)

## 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"> <li>• Create and configure HTTP destinations to make Web connections.</li> <li>• Connect to on-premise systems via HTTP, using the Cloud Connector.</li> </ul>

Protocol	Scenario
RFC	<p>Invoke on-premise ABAP function modules via RFC.</p> <ul style="list-style-type: none"> <li>• Create and configure RFC destinations.</li> <li>• Make connections to back-end systems via RFC, using the Cloud Connector.</li> </ul>
TCP	Access on-premise systems via TCP-based protocols using a SOCKS5 proxy.

Back to [Content](#)

## Restrictions

[General](#)

[Protocols](#)

[Cloud Foundry Environment](#)

[Cloud Connector](#)

### i Note

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

[General](#)

Topic	Restriction
<b>Java Connector</b>	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). On Windows platforms, you must install the <b>Microsoft Visual Studio C++ 2013</b> runtime libraries (vcredist_x64.exe), see <a href="#">Visual C++ Redistributable Packages for Visual Studio 2013</a> .
<b>Ports</b>	For Internet connections, you are allowed to use any port >1024. For cloud to on-premise solutions there are no port limitations.
<b>Destination Configuration</b>	<ul style="list-style-type: none"> <li>• You can use destination configuration files with extension .props, .properties, .jks, and .txt, as well as files with no extension.</li> <li>• If a destination configuration consists of a keystore or truststore, it must be stored in JKS files with a standard .jks extension.</li> </ul>

Back to [Restrictions](#)

[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](#)

## Cloud Foundry Environment

Topic	Restriction
Service Channels	Service channels are supported only for SAP HANA database, see <a href="#">Using Service Channels</a> .
E-Mail	E-mail functions are not supported.

Back to [Restrictions](#)

## Cloud Connector

Topic	Restriction
Scenarios	To learn in which system landscapes you can set up the Cloud Connector, see <a href="#">Extended Scenarios</a> .
Installation	To check all software and hardware restrictions for working with the Cloud Connector, see <a href="#">Prerequisites</a> .

Back to [Restrictions](#)

Back to [Content](#)

## Related Information

- [Connectivity in the Cloud Foundry Environment](#)
- [Cloud Connector](#)
- [Connectivity via Reverse Proxy](#)
- [Connectivity Support](#)
- [Connectivity Proxy for Kubernetes](#)
- [Transparent Proxy for Kubernetes](#)

## Cloud Connector

Learn more about the Cloud Connector: features, scenarios and setup.

### i 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

Basic Scenarios

Basic Tasks

Advantages

Extended Scenarios

What's New?

Please note that image maps are not interactive in PDF output.

### In this Guide

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

Configuration

Installation

Administration

FAQ

Upgrade

Security

Support

Update the Java VM

Security Guidelines

Uninstallation

Please note that image maps are not interactive in PDF output.

## 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](#)

## 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](#)

## Basic Scenarios

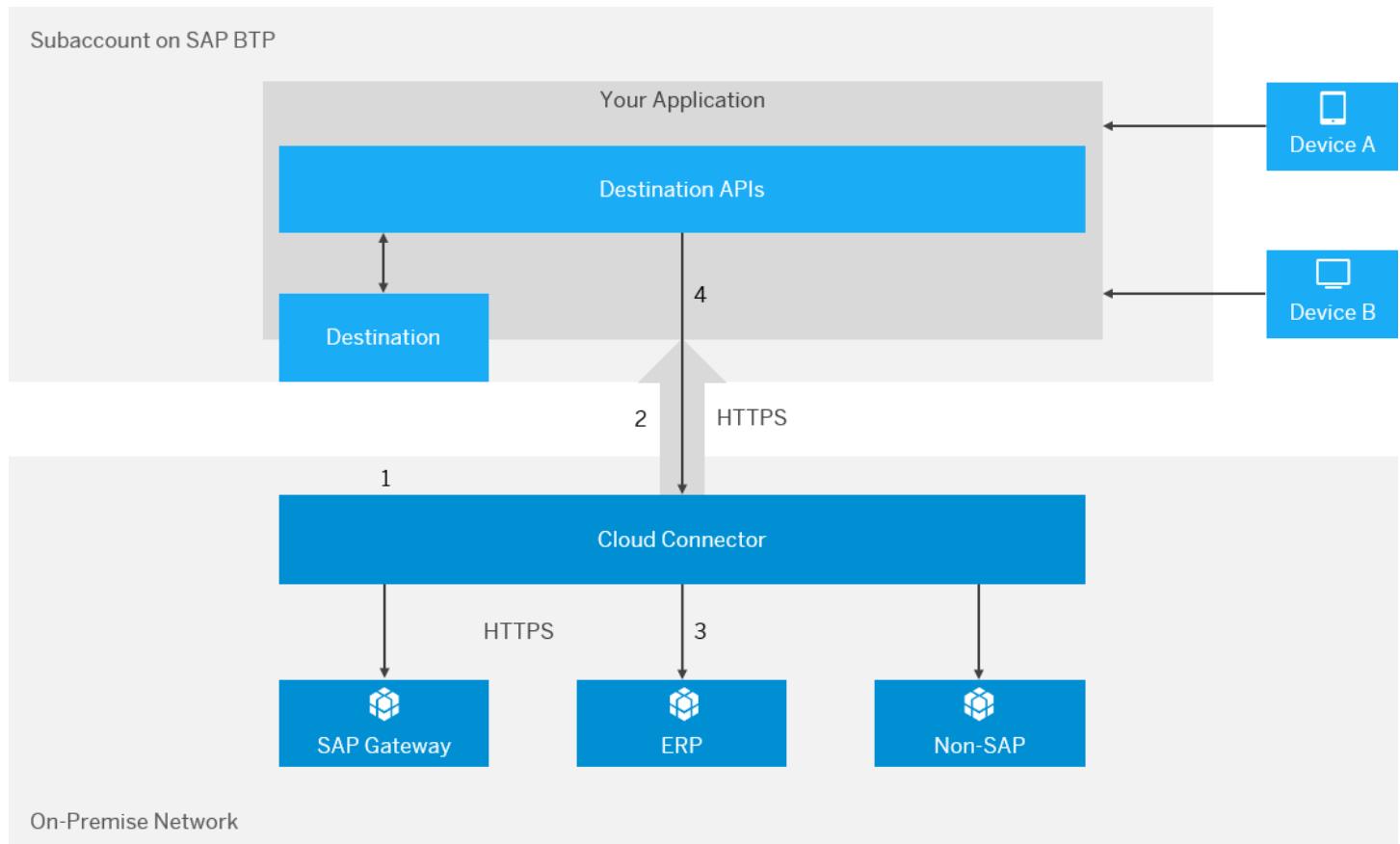
[Connecting Cloud Applications to On-Premise Systems](#)

[Connecting On-Premise Database Tools to SAP HANA Databases](#)

### 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](#).

[Connecting Cloud Applications to On-Premise Systems](#)



1. Install the Cloud Connector: [Installation](#)

2. Set up the connection between Cloud Connector, back-end system and your SAP BTP subaccount : [Initial Configuration](#), [Managing Subaccounts](#)

3. Allow your cloud application to access a back-end system on the intranet: [Configure Access Control](#)

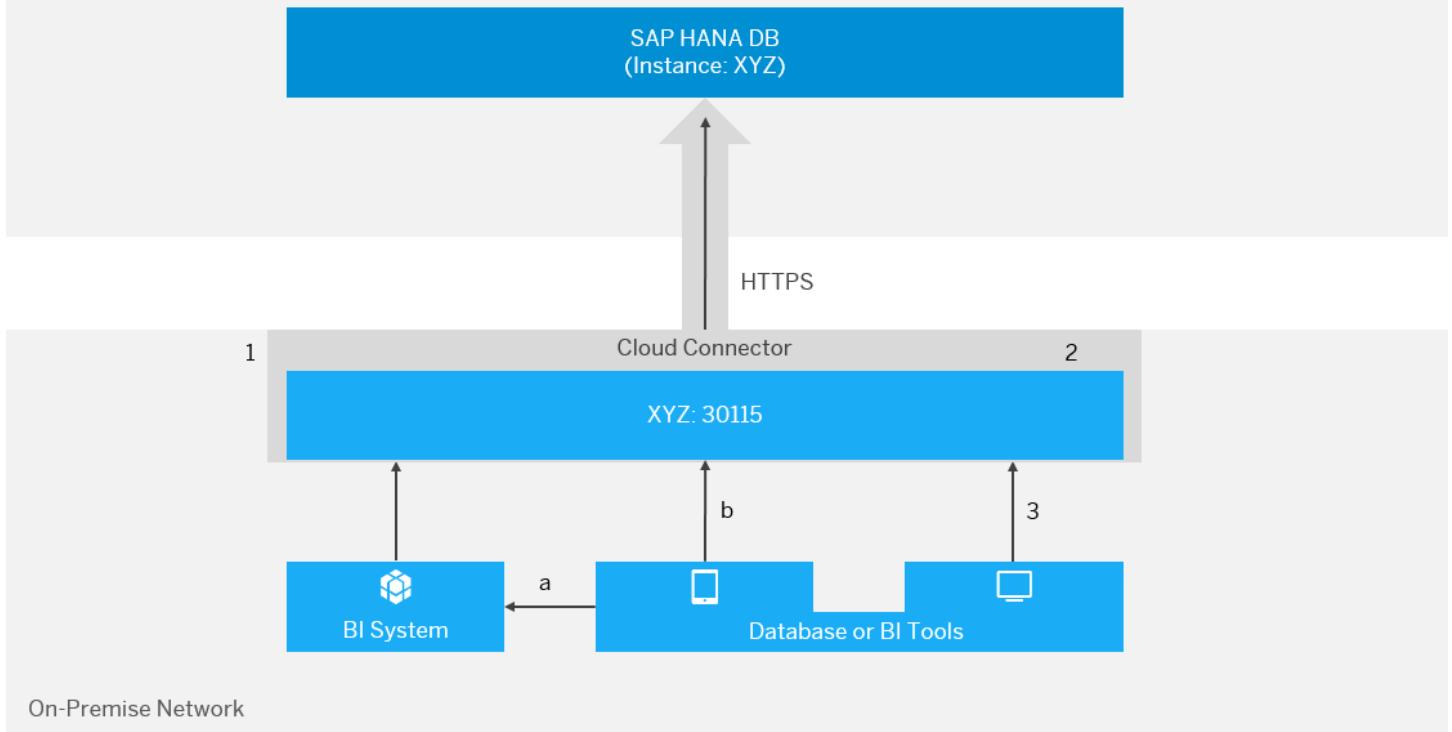
4. Connect your cloud application to an on-premise system:

[Consuming the Connectivity Service](#) (Cloud Foundry environment)

Back to [Basic Scenarios](#)

Back to [Content](#)

**Connecting On-Premise Database Tools to SAP HANA Databases**



1. Install and configure the Cloud Connector: [Installation](#), [Initial Configuration](#), [Managing Subaccounts](#)
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](#)

### i 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](#).

Back to [Basic Scenarios](#)

Back to [Content](#)

## 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
-----------------	-------------	----------

Cloud Connector	Environment	Examples
<b>Can</b> be set up in:	Customer on-premise network (see <a href="#">Basic Scenarios</a> )	SAP ERP, SAP S/4HANA
<b>i Note</b> 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.	SAP Hosting	SAP HANA Enterprise Cloud (HEC)
	Third-party <b>IaaS</b> providers (hosting)	Amazon Web Services (AWS), Microsoft Azure, Google Cloud
<b>Cannot</b> be set up in:	SAP <b>SaaS</b> solutions	SAP SuccessFactors, SAP Concur, SAP Ariba
	SAP cloud-based enterprise solutions	SAP S/4HANA Cloud, SAP C/4HANA
	Third-party <b>PaaS</b> providers	AWS, Azure, Google Cloud
	Third-party <b>SaaS</b> providers	

Back to [Content](#)

## Basic Tasks

The following steps are required to connect the Cloud Connector to your SAP BTP subaccount:

- Install the Cloud Connector: [Installation](#)
- Perform the initial configuration for the Cloud Connector: [Initial Configuration](#)
- Register the Cloud Connector for your SAP BTP subaccount: [Managing Subaccounts](#)

Back to [Content](#)

## What's New?

Follow the SAP BTP [Release Notes](#) to stay informed about Cloud Connector and Connectivity updates.

Back to [Content](#)

## Related Information

[Installation](#)

[Configuration](#)

[Administration](#)

[Security](#)

[Upgrade](#)

[Update the Java VM](#)

[Uninstallation](#)

[Frequently Asked Questions](#)

[REST APIs](#)

## 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.

### **i 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](#).
- For OS-specific requirements and procedures, see section **Tasks** below.

## Tasks

- [Installation on Microsoft Windows OS](#)
- [Installation on Linux OS](#)
- [Installation on Mac OS X](#)

## Related Information

[Sizing Recommendations](#)

[Recommendations for Secure Setup](#)

[Uninstallation](#)

# Prerequisites

Prerequisites for successful installation of the Cloud Connector.

## Content

Section	Description
<a href="#">Connectivity Restrictions</a>	General information about SAP BTP and connectivity restrictions.
<a href="#">Hardware</a>	Hardware prerequisites for a physical or virtual machine.
<a href="#">Software</a>	Required software download and installation.
<a href="#">JDKs</a>	Java Development Kit (JDK) versions that you can use.
<a href="#">Product Availability Matrix</a>	Availability of operating systems/versions for specific Cloud Connector versions.
<a href="#">Network</a>	Required Internet connection to SAP BTP hosts per region.

### i Note

For additional system requirements, see also [System Requirements](#).

## Connectivity Restrictions

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

For specific information about all Connectivity restrictions, see [Connectivity: Restrictions](#).

Back to [Content](#)

## 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
Memory (RAM)	2 GB	4 GB
Free disk space	3 GB	20 GB

Back to [Content](#)

## Software

- You have downloaded the Cloud Connector installation archive from [SAP Development Tools for Eclipse](#).
- A JDK 8 must be installed. You can download an up-to-date SAP JVM from [SAP Development Tools for Eclipse](#) as well.

### ⚠ 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 adopt the *server.xml* configuration file in directory <scc\_installation\_folder>/conf. To do so, follow the steps in [HTTPS port configuration](#) for APR.

Back to [Content](#)

## 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
<a href="#">SAP Machine</a> 64-bit	11	2.14.0 and higher
<a href="#">SAP Machine</a> 64-bit	17	2.15.0 and higher

### i Note

The support for using Cloud Connector with Java runtime version 7 ended on December 31, 2019. Any Cloud Connector version released after that date may contain Java byte code requiring at least a JVM 8.

We therefore strongly recommend that you perform **fresh** installations only with **Java 8**, and **update** existing installations running with Java 7, to Java 8.

See [SAP Cloud Connector – Java 7 support will phase out](#) and [Update the Java VM](#).

Back to [Content](#)

## 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

Operating System Version	Architecture	Cloud Connector Version
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	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

Back to [Content](#)

## 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, and Business Application Studio. Check the respective solution documentation for details.

### i 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](#)

[ABAP Environment](#)

[Neo Environment](#)

[Trial](#)

Region (Region Host)	Hosts	IP Address

Region (Region Host)	Hosts	IP Address
<b>Cloud Foundry Environment</b>		
<p><b>i Note</b></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 <a href="#">Regions</a>.</p>		
<i>Enterprise &amp; Trial</i> 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
	connectivitytunnel.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223
	connectivitytunnel.cf.eu10-002.hana.ondemand.com	3.64.227.236, 3.126.229.22, 18.193.180.19
	connectivitytunnel.cf.eu10-003.hana.ondemand.com	3.127.77.3, 3.64.196.58, 18.156.151.247
	connectivitytunnel.cf.eu10-004.hana.ondemand.com	3.65.185.47, 3.70.38.218, 18.196.206.8
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
	connectivitycertsigning.cf.eu11.hana.ondemand.com	3.124.207.41, 18.157.105.117, 18.156.209.198
	connectivitytunnel.cf.eu11.hana.ondemand.com	3.124.207.41, 18.157.105.117, 18.156.209.198
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
US 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

Region (Region Host)	Hosts	IP Address
US West (WA) - Azure (cf.us20.hana.ondemand.com)	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
	connectivitytunnel.cf.us10-001.hana.ondemand.com	3.220.114.17, 3.227.182.44, 52.86.131.53
	connectivitytunnel.cf.us10-002.hana.ondemand.com	34.202.68.0, 54.234.152.59, 107.20.66.86
US East (VA) - Azure (cf.us21.hana.ondemand.com)	connectivitynotification.cf.us20.hana.ondemand.com	40.91.120.100
	connectivitycertsigning.cf.us20.hana.ondemand.com	40.91.120.100
	connectivitytunnel.cf.us20.hana.ondemand.com	40.91.120.100
US Central (IA) - Google Cloud (cf.us30.hana.ondemand.com)	connectivitynotification.cf.us21.hana.ondemand.com	40.88.52.17
	connectivitycertsigning.cf.us21.hana.ondemand.com	40.88.52.17
	connectivitytunnel.cf.us21.hana.ondemand.com	40.88.52.17
Brazil (São Paulo) - AWS (cf.br10.hana.ondemand.com)	connectivitynotification.cf.us30.hana.ondemand.com	35.184.169.79
	connectivitycertsigning.cf.us30.hana.ondemand.com	35.184.169.79
	connectivitytunnel.cf.us30.hana.ondemand.com	35.184.169.79
Japan (Tokyo) - AWS (cf.jp10.hana.ondemand.com)	connectivitynotification.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4
	connectivitycertsigning.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4
	connectivitytunnel.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4
Japan (Tokyo) - Azure (cf.jp20.hana.ondemand.com)	connectivitynotification.cf.jp10.hana.ondemand.com	13.114.117.83, 3.114.248.68, 3.113.252.15
	connectivitycertsigning.cf.jp10.hana.ondemand.com	13.114.117.83, 3.114.248.68, 3.113.252.15
	connectivitytunnel.cf.jp10.hana.ondemand.com	13.114.117.83, 3.114.248.68, 3.113.252.15

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
	connectivitycertsigning.cf.ap10.hana.ondemand.com	13.236.220.84, 13.211.73.244, 3.105.95.184
	connectivitytunnel.cf.ap10.hana.ondemand.com	13.236.220.84, 13.211.73.244, 3.105.95.184
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
	connectivitycertsigning.cf.ap11.hana.ondemand.com	3.0.9.102, 18.140.39.70, 18.139.147.53
	connectivitytunnel.cf.ap11.hana.ondemand.com	3.0.9.102, 18.140.39.70, 18.139.147.53
Asia Pacific (Seoul) - AWS (cf.ap12.hana.ondemand.com)	connectivitynotification.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12
	connectivitycertsigning.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12
	connectivitytunnel.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12
Australia (Sydney) - Azure (cf.ap20.hana.ondemand.com)	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
Singapore - Azure (cf.ap21.hana.ondemand.com) <i>Enterprise &amp; Trial</i>	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
Canada (Montreal) - AWS (cf.ca10.hana.ondemand.com)	connectivitynotification.cf.ca10.hana.ondemand.com	35.182.75.101, 35.183.74.34
	connectivitycertsigning.cf.ca10.hana.ondemand.com	35.182.75.101, 35.183.74.34
	connectivitytunnel.cf.ca10.hana.ondemand.com	35.182.75.101, 35.183.74.34
Switzerland (Zurich) - Azure (cf.ch20.hana.ondemand.com)	connectivitynotification.cf.ch20.hana.ondemand.com	20.208.56.83
	connectivitycertsigning.cf.ch20.hana.ondemand.com	20.208.56.83
	connectivitytunnel.cf.ch20.hana.ondemand.com	20.208.56.83
India (Mumbai) - Google Cloud (cf.in30.hana.ondemand.com)	connectivitynotification.cf.in30.hana.ondemand.com	34.93.125.74
	connectivitycertsigning.cf.in30.hana.ondemand.com	34.93.125.74

Region (Region Host)	Hosts	IP Address
	connectivitytunnel.cf.in30.hana.ondemand.com	34.93.125.74
China (Shanghai) - Alibaba Cloud (cf.cn40.platform.sapcloud.cn)	connectivitynotification.cf.cn40.platform.sapcloud.cn	139.224.7.71
	connectivitycertsigning.cf.cn40.platform.sapcloud.cn	139.224.7.71
	connectivitytunnel.cf.cn40.platform.sapcloud.cn	139.224.7.71

[Back to Network](#)[Back to Content](#)

Region (Region Host)	Hosts	IP Address
----------------------	-------	------------

**ABAP Environment****i Note**

For scenarios using the ABAP environment, include the IPs of the corresponding region below in your firewall rules, **in addition** to the IPs listed for the same region above (Cloud Foundry environment), if you use IP-based firewall rules.

Europe (Frankfurt) - AWS	*.abap.eu10.hana.ondemand.com	18.185.129.45, 52.29.97.247
US East (VA) - AWS	*.abap.us10.hana.ondemand.com	52.4.22.116, 52.1.251.254
Japan (Tokyo) - AWS	*.abap.jp10.hana.ondemand.com	18.177.215.21, 3.115.146.41

[Back to Network](#)[Back to Content](#)

Region (Region Host)	Hosts	IP Address
----------------------	-------	------------

**Neo Environment**

Europe (Rot) (hana.ondemand.com) (eu1.hana.ondemand.com)	connectivitynotification.hana.ondemand.com	155.56.210.83
	connectivitycertsigning.hana.ondemand.com	155.56.210.43
	connectivitytunnel.hana.ondemand.com	155.56.210.84
Europe (Frankfurt) (eu2.hana.ondemand.com)	connectivitynotification.eu2.hana.ondemand.com	157.133.206.143
	connectivitycertsigning.eu2.hana.ondemand.com	157.133.205.174
	connectivitytunnel.eu2.hana.ondemand.com	157.133.205.233
Europe (Amsterdam) (eu3.hana.ondemand.com)	connectivitynotification.eu3.hana.ondemand.com	130.214.87.72
	connectivitycertsigning.eu3.hana.ondemand.com	130.214.87.76
	connectivitytunnel.eu3.hana.ondemand.com	130.214.86.212
United States East (Ashburn) (us1.hana.ondemand.com)	connectivitynotification.us1.hana.ondemand.com	130.214.181.204
	connectivitycertsigning.us1.hana.ondemand.com	130.214.181.48
	connectivitytunnel.us1.hana.ondemand.com	130.214.181.134
United States West (Chandler) (us2.hana.ondemand.com)	connectivitynotification.us2.hana.ondemand.com	130.214.129.127
	connectivitycertsigning.us2.hana.ondemand.com	130.214.129.140

Region (Region Host)	Hosts	IP Address
	connectivitytunnel.us2.hana.ondemand.com	130.214.129.88
United States East (Sterling) (us3.hana.ondemand.com)	connectivitynotification.us3.hana.ondemand.com	130.214.32.144
	connectivitycertsigning.us3.hana.ondemand.com	130.214.33.92
	connectivitytunnel.us3.hana.ondemand.com	130.214.33.119
US States West (Colorado Springs) (us4.hana.ondemand.com)	connectivitynotification.us4.hana.ondemand.com	130.214.183.46
	connectivitycertsigning.us4.hana.ondemand.com	130.214.183.14
	connectivitytunnel.us4.hana.ondemand.com	130.214.183.103
Australia (Sydney) (ap1.hana.ondemand.com)	connectivitynotification.ap1.hana.ondemand.com	157.133.97.47
	connectivitycertsigning.ap1.hana.ondemand.com	157.133.97.27
	connectivitytunnel.ap1.hana.ondemand.com	157.133.97.46
China (Shanghai) (cn1.platform.sapcloud.cn)	connectivitynotification.cn1.platform.sapcloud.cn	121.91.109.81
	connectivitycertsigning.cn1.platform.sapcloud.cn	121.91.109.77
	connectivitytunnel.cn1.platform.sapcloud.cn	121.91.109.82
Japan (Tokyo) (jp1.hana.ondemand.com)	connectivitynotification.jp1.hana.ondemand.com	130.214.112.168
	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 <a href="#">Network</a>		
Back to <a href="#">Content</a>		
Region (Region Host)	Hosts	IP Address

Region (Region Host)	Hosts	IP Address
<b>Trial (Cloud Foundry Environment)</b>		
<p><b>i Note</b></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 <a href="#">Regions</a>.</p>		
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
	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](#)

Back to [Content](#)

### **i 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](#).

For more information about the used ABAP server ports, see: [Ports of SAP NetWeaver Application Server ABAP](#).

[Back to Network](#)

[Back to Content](#)

## Related Information

[Installation on Microsoft Windows OS](#)

[Installation on Linux OS](#)

[Installation on Mac OS X](#)

[Recommendations for Secure Setup](#)

[Sizing Recommendations](#)

## 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 <scc\_dir>/log/ within the Cloud Connector *root* directory. The *js\_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.

### i 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](#).

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](#)

# 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.

### i 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.

# 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](#).

### i 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.

### i 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](#), 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](#)

[Configuration Setup](#)

## 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.

### i 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](#)

[Sizing for the Shadow Instance](#)

## 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.

### i 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
<p><b>S:</b></p> <p>The expected load is small (up to <b>1 million requests</b> per day, request concurrency and size is in average low) and only a <b>few subaccounts with some applications</b> are connected.</p> <p>In addition, only a <b>few service channels</b> are used, only <b>small data amount is replicated</b> to cloud systems.</p> <p>Setup can be done in a virtual or physical machine.</p>	2 cores 2.6 GHz	4GB RAM / 1GB / 2GB	10GB
<p><b>M :</b></p> <p>The expected load is medium (up to <b>10 million requests</b> per day, request concurrency and size is in average medium) and <b>multiple subaccounts with multiple applications</b> are connected.</p> <p>In addition, <b>many service channels</b> are used, and a <b>medium data amount is replicated</b> to cloud systems.</p> <p>We recommend that you do the setup in a virtual or physical machine. A virtual machine should be on a host without overcommitted resources.</p>	4 cores 3.0 Ghz	16GB RAM / 4GB / 8GB	20GB
<p><b>L:</b></p> <p>The expected load is large (<b>more than 10 million requests</b> per day, request concurrency and size is in average medium or high) and multiple subaccounts with multiple applications are connected.</p> <p>In addition, many service channels are used, and a <b>large data amount is replicated</b> to cloud systems concurrently.</p> <p>We recommend that you do the setup in a virtual or physical machine. A virtual machine should be on a host without overcommitted resources.</p>	8 cores 3.0 Ghz	32GB RAM / 8GB / 16GB	40GB

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](#).

### **i Note**

You should use the same value for both *<initial heap size>* and *<maximum heap size>*.

## 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.

### **i 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

## 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](#)). For S installations, the default configuration will probably be sufficient to handle the traffic.

To change the connection parameters, proceed as follows:

- As of Cloud Connector 2.11, you can configure the number of physical connections through the Cloud Connector UI. See also [Configure Advanced Connectivity](#).
- 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.

### **i 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](#)

[On-Premise To On-Demand Connections \(Service Channels\)](#)

## 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.

### **i 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.

### i 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.

## 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](#). 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.

## 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](#).
- 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 vcredist\_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](#).

### **i 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](#).

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.

### **i 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.

### **i 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](#).

## Related Information

[Recommendations for Secure Setup](#)

## 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 -xzof" 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](#).
- 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](#).

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 -xzof sapcc-<version>-linux-<platform>.tar.gz
```

### i 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.

### i 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 (as of Cloud Connector version 2.12.3)

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. Make sure `JAVA_HOME` is set to the JVM used. Then execute the following command to reinstall the daemon:

```
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](#).

## Related Information

[Recommendations for Secure Setup](#)

## Installation on Mac OS X

Installing the Cloud Connector on a Mac OS X operating system.

### Prerequisites

**i Note**

Mac OS X is not supported for productive scenarios. The developer version described below must not be used as productive version.

**⚠ Caution**

The Cloud Connector does not natively support the M1/M2 architecture yet. Make sure you use a JVM based on the `x64` CPU architecture, and not the `aarch64` one.

- You have one of the supported 64-bit operating systems. For more information, see [Product Availability Matrix](#).
- You have downloaded the `tar .gz` archive for the developer use case on Mac OS X from [SAP Development Tools for Eclipse](#).
- A supported Java version must be installed. For more information, see [JDKs](#).  
If 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 -xzof sapcc-<version>-macosx-x64.tar.gz
```

2. Go to this directory and start Cloud Connector using the `go .sh` script.

3. Continue with the `Next Steps` section.

**i Note**

The Cloud connector is not started as a daemon, and therefore will not automatically start after a reboot of your system. Also, the Mac OS X version of 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](#).

## Related Information

### [Recommendations for Secure Setup](#)

## 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 top navigation bar includes the SAP logo, the title "Cloud Connector Administration", and a user dropdown for "Administrator". The left sidebar has a "Connector" section with a dropdown menu showing "Subaccount: conn2" and options like "Security Status", "Alerting", "High Availability", "Hardware Metrics Monitor", and "Configuration". Below this is another section with "conn2" and links to "Cloud To On-Premise", "On-Premise To Cloud", "Monitor", "Audits", and "Log And Trace Files". The main content area is titled "Security Status" and contains a "Risk" section with a red exclamation mark icon. Below this is a "General Security Status" table:

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	✎

Below the general status is a "Subaccount-Specific Security Status" table:

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.

### i 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](#) 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.

### **i 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.

### **i 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 Authentication](#).

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](#).

### **i 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).

## i 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 and RFC 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 SSL-encrypted or SNC-encrypted. See [Configure Access Control \(HTTP\)](#) and [Configure Access Control \(RFC\)](#).

## 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 SAP Cloud Connector configuration interface. On the left, there's a sidebar with various monitoring and connectivity options like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. The main area is titled 'Configuration' and has tabs for 'USER INTERFACE', 'CLOUD', 'ON PREMISE', 'REPORTING', and 'ADVANCED'. Under 'USER INTERFACE', the 'Authentication Mode' is set to 'Password' and the 'User Name' is 'Administrator'. The 'UI Certificate' section displays certificate details: Subject DN (CN=SCC,OU=Connectivity,O=SAP SE,C=DE), Issuer (CN=SCC,OU=Connectivity,O=SAP SE,C=DE), Valid From (2022-08-23 19:55:27 +0200), and Valid To (2024-12-12 18:55:27 +0100). Below this is a 'Certificates Chain' section with a subject DN of CN=SCC,OU=Connectivity,O=SAP SE,C=DE. A large red box highlights the 'Cipher Suites (45)' table, which lists 12 cipher suites with columns for Status Quo, Status New, Security, and Name. The table includes rows for TLS\_AES\_128\_GCM\_SHA256, TLS\_AES\_256\_GCM\_SHA384, TLS\_DHE\_DSS\_WITH\_AES\_128\_CBC\_SHA, TLS\_DHE\_DSS\_WITH\_AES\_128\_GCM\_SHA256, TLS\_DHE\_DSS\_WITH\_AES\_256\_CBC\_SHA, TLS\_DHE\_DSS\_WITH\_AES\_256\_GCM\_SHA256, TLS\_DHE\_DSS\_WITH\_AES\_256\_CBC\_SHA256, TLS\_DHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256, TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA, and 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.

### i 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](#)

[Security](#)

## Configuration

Configure the Cloud Connector to make it operational for connections between your SAP BTP applications and on-premise systems.

Topic	Description
-------	-------------

Topic	Description
<a href="#">Initial Configuration</a>	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.
<a href="#">Managing Subaccounts</a>	How to connect SAP BTP subaccounts to your Cloud Connector.
<a href="#">Authenticating Users against On-Premise Systems</a>	Basic authentication and principal propagation (user propagation) are the authentication types currently supported by the Cloud Connector.
<a href="#">Configure Access Control</a>	Configure access control or copy the complete access control settings from another subaccount on the same Cloud Connector.
<a href="#">Configuration REST APIs</a>	Configure a newly installed Cloud Connector (initial configuration, subaccounts, access control) using the configuration REST API.
<a href="#">Using Service Channels</a>	Service channels provide access from an external network to certain services on SAP BTP, which are not exposed to direct access from the Internet.
<a href="#">Configure Trust</a>	Set up an allowlist for trusted cloud applications and a trust store for on-premise systems in the Cloud Connector.
<a href="#">Connect DB Tools to SAP HANA via Service Channels</a>	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.
<a href="#">Configure Domain Mappings for Cookies</a>	Map virtual and internal domains to ensure correct handling of cookies in client/server communication.
<a href="#">Configure Solution Management Integration</a>	Activate Solution Management reporting in the Cloud Connector.
<a href="#">Configure Advanced Connectivity</a>	Adapt connectivity settings that control the throughput and HTTP connectivity to on-premise systems.
<a href="#">Configure the Java VM</a>	Adapt the JVM settings that control memory management.
<a href="#">Configuration Backup</a>	Backup and restore your Cloud Connector configuration.
<a href="#">Configure Login Screen Information</a>	Add additional information to the login screen and configure its appearance.

## 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](#)

#### [Log on to the Cloud Connector](#)

#### [Change your Password and Choose Installation Type](#)

#### [Set up Connection Parameters and HTTPS Proxy](#)

#### [Establish Connections to SAP BTP](#)

## Prerequisites

- You have downloaded and installed the Cloud Connector, see [Installation](#).
- 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:

### i 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
<b>Cloud Foundry</b> [feature set A]	The user must be a <i>member of the global account</i> that the subaccount belongs to.  Alternatively, you can assign the user as <i>Security Administrator</i> .	<a href="#">Add Members to Your Global Account</a>  <a href="#">Managing Security Administrators in Your Subaccount [Feature Set A]</a>
<b>Cloud Foundry</b> [feature set B]	Assign at least one of these <i>default role collections</i> (all of them including the role <b>Cloud Connector Administrator</b> ): <ul style="list-style-type: none"> <li>◦ Subaccount Administrator</li> <li>◦ Cloud Connector Administrator</li> <li>◦ Connectivity and Destination Administrator</li> </ul> Alternatively, you can assign a <i>custom role collection</i> to the user that includes the role <b>Cloud Connector Administrator</b> .	<a href="#">Default Role Collections [Feature Set B]</a>  <a href="#">Role Collections and Roles in Global Accounts, Directories, and Subaccounts [Feature Set B]</a>
<b>Neo</b>	Assign at least one of these <i>default roles</i> : <ul style="list-style-type: none"> <li>◦ Cloud Connector Admin</li> <li>◦ Administrator</li> </ul> Alternatively, you can assign a <i>custom role</i> to the user that includes the permission <code>manageSCCTunnels</code> .	<a href="#">Managing Member Authorizations in the Neo Environment</a>

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).

### i 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](#). For operating the Cloud Connector securely, see also [Security Guidelines](#).

## 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>`

- o `<hostname>` refers to the machine on which the Cloud Connector is installed. If installed on your machine, you can simply enter `localhost`.
- o `<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>`.

## Change your Password and Choose Installation Type

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](#).

**Mandatory Password Change**

Current Password:

New Password:

Repeat New Password:

**Choose Installation Type**

Master (Primary Installation)

Shadow (Backup Installation)

Description:

3. You can edit the password for the **Administrator** user from **Configuration** in the main menu, tab **User Interface**, section **Authentication**:

**Edit Authentication**

User Name:

Current Password\*:

New Password:

Repeat New Password:

**UI Certificate**

Subject DN: CN=SCC, OU=

Issuer: CN=SCC, OU=

Valid From: 02 March 2015

Valid To: 12 October 2015

**Save Cancel**

### i 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](#)

## 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:

If your internal landscape is protected by a firewall that blocks any outgoing TCP traffic, you must specify an HTTPS proxy that the Cloud Connector can use to connect to SAP BTP. Normally, you must use the same proxy settings as those being used by your standard Web browser. The Cloud Connector needs this proxy for two operations:

- Download the correct connection configuration corresponding to your subaccount ID in SAP BTP.
- Establish the SSL tunnel connection from the Cloud Connector user to your SAP BTP subaccount.

### i 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](#).

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](#).

### i 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](#).

2. For *<Subaccount>*, *<Subaccount User>* and *<Password>*, enter the values you obtained when you registered your subaccount on SAP BTP.

### i 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 \(Cloud Foundry Environment\)](#). 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.

### **i 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](#).

3. (Optional) You can define a *<Display Name>* that lets you easily recognize a specific subaccount in the UI compared to the technical subaccount name.
4. (Optional) You can define a *<Location ID>* identifying the location of this Cloud Connector for a specific subaccount. As of Cloud Connector release **2.9.0**, 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 URL. To route requests to a Cloud Connector with a location ID, the location ID must be configured in the respective destinations.

### **i Note**

Location IDs provided in older versions of the Cloud Connector are discarded during upgrade to ensure compatibility for existing scenarios.

5. Enter a suitable proxy host from your network and the port that is specified for this proxy. If your network requires an authentication for the proxy, enter a corresponding proxy user and password. You must specify a proxy server that supports SSL communication (a standard HTTP proxy does not suffice).

### **i Note**

These settings strongly depend on your specific network setup. If you need more detailed information, please contact your local system administrator.

6. (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 SSL 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 back-end systems. To allow your on-demand applications to access specific internal back-end systems, proceed with the access configuration described in the next section.

## i 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.

The screenshot shows the SAP Cloud Connector Administration interface. The top navigation bar includes the SAP logo, 'Cloud Connector Administration', and a user 'Administrator'. The left sidebar has sections like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration' (which is selected). Under 'Configuration', there's a dropdown for 'conn2' with options like 'Cloud To On-Premise', 'On-Premise To Cloud', 'Monitor', 'Audits', and 'Log And Trace Files'. The main content area has tabs for 'USER INTERFACE', 'CLOUD' (which is selected), 'ON PREMISE', 'REPORTING', and 'ADVANCED'. Under 'CLOUD', there's a 'Connector Info' section with a 'Description:' field and a 'HTTPS Proxy' section. The 'HTTPS Proxy' section is highlighted with a red border and contains fields for 'Host:', 'Port:', and 'User:'. There are edit, info, and help icons for each section.

- 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:

This screenshot shows the same SAP Cloud Connector Administration interface as the previous one, but the 'Connector Info' section in the 'Cloud' tab now has a 'Description:' field containing the value 'Cloud Connector on VM in DMZ'. The rest of the interface is identical to the first screenshot.

[Back to Tasks](#)

## 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](#).

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 screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a sidebar with options like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. The 'Trial' subaccount is selected. The main area shows 'Subaccount Overview' with details: Region: Europe (Rot) - Trial, Region Host: hanatrial.ondemand.com (with a green icon), HTTPS Proxy: (grey diamond icon), Subaccount Certificate: Certificate valid until 12 August 2021 16:25:36 CEST (green icon), System Certificate: (grey diamond icon). Buttons at the top right include 'Disconnect', 'Import', 'Export', and 'Certificate'.

- 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](#).

## i 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](#)

## Related Information

- [Managing Subaccounts](#)
- [Initial Configuration \(HTTP\)](#)
- [Initial Configuration \(RFC\)](#)
- [Configuring the Cloud Connector for LDAP](#)
- [Managing Member Authorizations in the Neo Environment](#)
- [Use a Custom IDP for Subaccount Configuration](#)

## Initial Configuration (HTTP)

Configure the Cloud Connector for HTTP communication.

## Installation of 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 CA that signed the Cloud Connector's client 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:

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar has sections like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. Under 'Configuration', there are sub-sections: 'Cloud To On-Premise', 'On-Premise To Cloud', 'Monitor', 'Audits', and 'Log And Trace Files'. The main area has tabs: 'USER INTERFACE', 'CLOUD', 'ON PREMISE' (which is selected), 'REPORTING', and 'ADVANCED'. In the 'ON PREMISE' tab, there is a 'System Certificate' section. A modal dialog box titled 'Import System Certificate' is open. It contains fields for 'Subject DN:', 'Issuer:', 'Valid From:', and 'Valid To:'. Under 'Trust Store (0)', it shows 'Status' and 'X.509 Certificate'. The 'P12 Certificate:' field has a red border, indicating it is required. Below it is a 'Password:' field. At the bottom of the dialog are 'Import' and 'Cancel' buttons. The background shows a table with columns for actions like edit, delete, and info.

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](#) and upload the resulting signed certificate.

The screenshot shows the 'System Certificate' table. It has columns for 'Subject DN:', 'Issuer:', 'Valid From:', and 'Valid To:'. To the right of the table is a set of icons for actions: edit, delete, up, down, copy, paste, info, and help. Below the table is a 'Subject Alternative Names' section with a table header 'Type' and 'Value'. The table body shows 'No data'.

As of version 2.10, there is a third option - generating 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](#):

The screenshot shows the 'System Certificate' table. It has columns for 'Subject DN:', 'Issuer:', 'Valid From:', and 'Valid To:'. To the right of the table is a set of icons for actions: edit, delete, up, down, copy, paste, info, and help. Below the table is a 'Subject Alternative Names' section with a table header 'Type' and 'Value'. The table body shows 'No data'.

If a system certificate has been imported successfully, its distinguished name, the name of the issuer, and the validity dates are displayed:

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a sidebar with options like Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, and Configuration (which is selected). Below that is a sub-sidebar for 'conn2' with options like Cloud To On-Premise, On-Premise To Cloud, Monitor, Audits, and Log And Trace Files. The main content area has tabs for USER INTERFACE, CLOUD, ON PREMISE (which is active), REPORTING, and ADVANCED. Under the ON PREMISE tab, there's a 'System Certificate' section. It shows the following details:

Subject DN:	CN=Hugo, OU=CSI, O=SAP Trust Community, C=DE	Subject Alternative Names	
Issuer:	CN=SAP DigSig TestCA, O=SAP Trust Community Test, C=DE	Type	Value
Valid From:	16 March 2016 13:01:53 CET	No data	
Valid To:	16 March 2017 13:01:53 CET		

If a system certificate is no longer required, you can delete it. To do this, use the respective button and confirm deletion. If you need the public key for establishing trust with a server, you can simply export the full chain via the [Export](#) button.

## Related Information

[Configure Access Control \(HTTP\)](#)

## 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 Principal Propagation for RFC](#).

### 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:**

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a navigation tree with 'Connector' selected. Under 'Connector', 'Configuration' is also selected. In the main area, 'Subaccount: conn2' is shown. Below it, tabs for 'USER INTERFACE', 'CLOUD', 'ON PREMISE', 'REPORTING', and 'ADVANCED' are present. The 'ON PREMISE' tab is active. A modal window titled 'Edit SNC' is overlaid on the page. It contains three input fields: 'Library' with the value '/opt/sap/security/snc/libsapcrypto.so', 'My Name' with the value 'p:CN=SCC, O=MYCOMPANY, C=DE', and 'Quality of Protection' set to 'Default Protection'. At the bottom of the modal are 'Save' and 'Cancel' buttons.

- **Library Name:** Provides the location of the SNC library you are using for the Cloud Connector.

### i 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.

### i 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

### i 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.

### i 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 `sapcrypto.lib`).
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.

a. For Linux, you can set it solely for the Cloud Connector process by extending the daemon as described in [Installation on Linux OS](#).

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](#) and, if needed, [Configure Principal Propagation for RFC](#).

If you have further issues, check SAP note [1525059](#).

## Related Information

[Configure Principal Propagation for RFC](#)

# 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](#)

[Initial Configuration](#)

## 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 \(LDAP\)](#).

**Scenario 2:** Internal Cloud Connector user management. Activate LDAP user management in the Cloud Connector: [Use LDAP for Authentication](#).

# 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](#)

[Get the URL \[Feature Set A\]](#)

[Get the URL \[Feature Set B\]](#)[Get the One-Time Passcode](#)

## 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
<a href="#">Set up Connection Parameters and HTTPS Proxy</a>	Add an <i>initial</i> subaccount to a fresh Cloud Connector installation.
<a href="#">Managing Subaccounts</a>	Add <i>additional</i> subaccounts to an existing Cloud Connector installation.
<a href="#">Update the Certificate for a Subaccount</a>	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](#)):

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a navigation sidebar with options like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', 'Configuration', 'Define Subaccount' (which is currently selected), 'Cloud To On-Premise', 'On-Premise To Cloud', 'Monitor', 'Audits', and 'Log And Trace Files'. The main content area has a title 'Define Subaccount' and a message box stating 'Cloud Connector is not configured and remains inoperative unless you define at least one subaccount'. Below this, there are two sections: 'First Subaccount' and 'HTTPS Proxy'. In the 'First Subaccount' section, there are fields for 'Region:', 'Subaccount:', 'Display Name:', 'Subaccount User:' (containing '\$SAP-CP-SSO-PASSCODE\$'), 'Password:' (containing '<Passcode>'), 'Location ID:' (with placeholder 'Enter location ID to overwrite default'), and 'Description:'. A red box highlights the 'Subaccount User:' field. The 'HTTPS Proxy' section has fields for 'Host:', 'Port:', 'User:', and 'Password:'.

- When adding one ore more **additional subaccount(s)** to an existing Cloud Connector installation, provide the user name **\$SAP-CP-SSO-PASSCODE\$** and the passcode via the **Connector** screen (see also [Managing Subaccounts](#)):

SAP Cloud Connector Administration

Connector Subaccount: trial

**Connector**

+ Add Subaccount Backup ...

**Connector Overview**

Connector ID: ..... Security Status: Low risk  
Local Name: ..... High Availability: Disconnected  
Local IP: ..... Alerts: 19

## Add Subaccount

Region:\*

Subaccount:\*

Display Name:

Subaccount User:\*

\$SAP-CP-SSO-PASSCODE\$

Password:\*

<Passcode>

Location ID: Enter location ID to overwrite default

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](#)):

SAP Cloud Connector Administration

Connector Subaccount: trial

**trial** Disconnected

**Subaccount Overview**

Region: Canary CF (Fraud) Region Host: cf.sap.hana.eu Subaccount Certificate: Certificate has been updated System Certificate: CN=\*.wdf.sap.corp

Refresh Certificate

User Name: \$SAP-CP-SSO-PASSCODE\$  
Password: <Passcode>

Refresh Cancel

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.

## i 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 \[Feature Set A\]](#)

[Get the URL \[Feature Set B\]](#)

## ⚠ Caution

Mind the respective user rights described in the prerequisites for [Initial Configuration](#) and [Managing Subaccounts](#). For feature set A, it is mandatory to be a subaccount *Security Administrator*, not only a global account member.

Back to [Content](#)

## 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](#)

Back to [Content](#)

## 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-XSUAH-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:
  - o [Create and Bind a Connectivity Service Instance](#)
  - o [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](#)

Back to [Content](#)

## Get the One-Time Passcode

1. Open the resulting URL in your browser to get the one-time passcode via SSO:
  - o If *there is* an active user session, the passcode is generated automatically and returned right away.
  - o 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](#)

## Managing Subaccounts

Add and connect your SAP BTP subaccounts to the Cloud Connector.

### i 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.

### i 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:

### i 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]	The user must be a <i>member of the global account</i> that the subaccount belongs to.  Alternatively, you can assign the user as <i>Security Administrator</i> .	<a href="#">Add Members to Your Global Account</a>  <a href="#">Managing Security Administrators in Your Subaccount [Feature Set A]</a>
Cloud Foundry [feature set B]	Assign at least one of these <i>default role collections</i> (all of them including the role <b>Cloud Connector Administrator</b> ): <ul style="list-style-type: none"> <li>• Subaccount Administrator</li> <li>• Cloud Connector Administrator</li> <li>• Connectivity and Destination Administrator</li> </ul> Alternatively, you can assign a <i>custom role collection</i> to the user that includes the role <b>Cloud Connector Administrator</b> .	<a href="#">Default Role Collections [Feature Set B]</a>  <a href="#">Role Collections and Roles in Global Accounts, Directories, and Subaccounts [Feature Set B]</a>

Environment	Required Roles/Role Collections	More Information
Neo	<p>Assign at least one of these <i>default roles</i>:</p> <ul style="list-style-type: none"> <li>Cloud Connector Admin</li> <li>Administrator</li> </ul> <p>Alternatively, you can assign a <i>custom role</i> to the user that includes the permission <code>manageSCCTunnels</code>.</p>	<a href="#">Managing Member Authorizations in the Neo Environment</a>

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.

### i 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.

Status	Subaccount	Display Name	Location ID	Region	Actions
OK	819986f5-460d-4a36-b...	qual	EMEA	Europe (Frankfurt) - AWS	>
WARN	ab1220dc-5fb-45d1-9...	trial		Trial	>
OK	ztglf9vgj9	Virtual Machine Test	EMEA	Trial	>

In the screenshot above, the subaccount with display name *trial* (actual subaccount ID starts with **ab1220dc-5fb-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 [filter buttons](#) above the dashboard let you 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](#) 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="text"/>	
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](#).

2. For <Subaccount> and <Subaccount User> (user/password), enter the values you obtained when you registered your account on SAP BTP.

### i 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 \(Cloud Foundry Environment\)](#). 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.

### i 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](#).

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. As of Cloud Connector release 2.9.0, 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](#)
- [Update the Certificate for a Subaccount](#)
- [Configure a Disaster Recovery Subaccount](#)
- [Find Your Subaccount ID \(Cloud Foundry Environment\)](#)
- [Configure Custom Regions](#)

## 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 Coud** sections to a new subaccount, by using the export and import functions in the Cloud Connector administration UI.

### i 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.

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar has a tree view with nodes like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. Under 'Connector', 'trial' is selected. The main area shows a table with one row for 'trial'. The row contains fields: Region (Europe (Rot) - Trial), Region Host (hanatrial.ondemand.com), HTTPS Proxy (checkbox), Subaccount Certificate (checkbox, valid until 13 August 2021 15:07:54 CEST), System Certificate (checkbox). Below the table is a 'Subaccount Overview' section with fields: Subaccount (trial), Initiated By (Anonymous), Location ID (empty), Description (empty). The top right has buttons for 'Disconnect', 'Import' (with an upward arrow icon), 'Export' (with a downward arrow icon, highlighted with a red box), and 'Certificate'.

### 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.

This screenshot is identical to the previous one, showing the SAP Cloud Connector Administration UI with the 'trial' subaccount selected. The 'Import' button in the top right corner is highlighted with a red box.

3. Select one of the following sources:

- a. **File**, if you want to copy the configuration from a previously downloaded zip file.
- b. **Subaccount**, if you want to copy the configuration directly from another existing subaccount.

**Import Account Configuration**

Source:  File  
 Subaccount

File:

## 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](#) (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.

#### i 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.

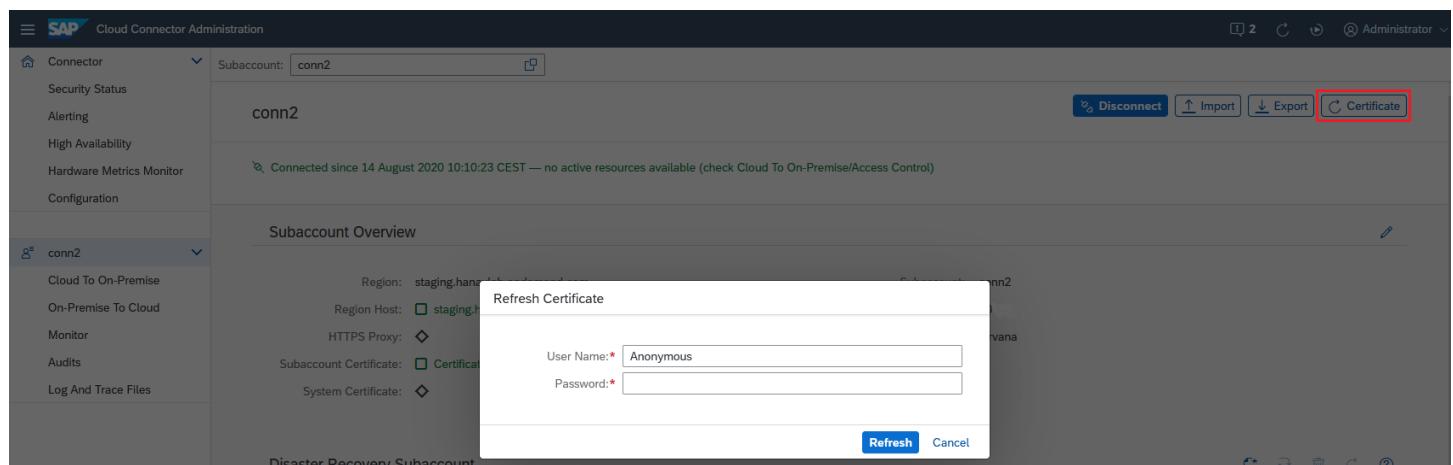
#### i Note

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**.



# Configure a Disaster Recovery Subaccount

Configure a subaccount as backup for disaster recovery.

## ⚠ Caution

This feature is deprecated. 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.

## i 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**.

### i Note

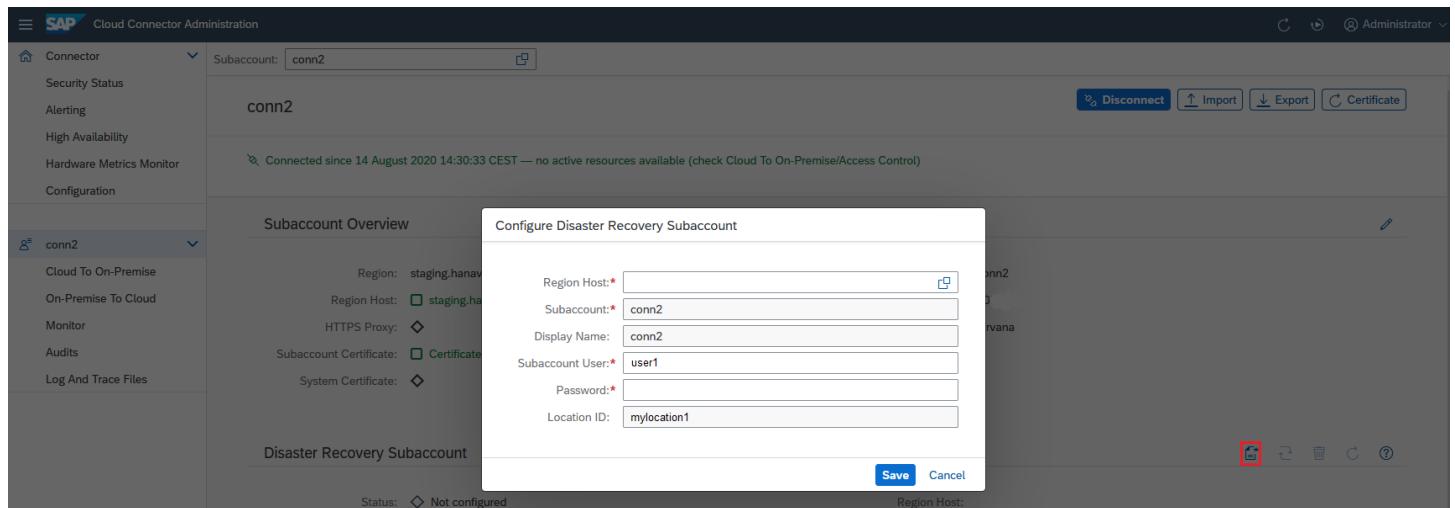
The technical subaccount name, the display name, and the location ID must remain the same. They are set automatically and cannot be changed.

### i Note

You cannot choose another original subaccount nor a trial subaccount to become a disaster recovery subaccount.

### i 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.

### i 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](#)

## 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.

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**.

The screenshot shows the SAP Cloud Connector Administration interface. At the top, it says 'Cloud Connector Administration' with a subaccount dropdown set to 'Testing'. Below that, it displays a 'Disaster Recovery Subaccount: Testing' with a note that it was 'Connected since Apr 10, 2019 3:28:39 PM — no active resources available (check Cloud To On-Premise/Access Control)'. There are buttons for 'Disconnect', 'Import', 'Export', and 'Certificate'. The main area is divided into 'Connector State' and 'Original Subaccount'. In 'Connector State', fields include Region, Subaccount, Initiated By, Location ID, and Description, all with placeholder values. In 'Original Subaccount', fields include Status (Connected), Region Host, Subaccount, and User, also with placeholder values. A red box highlights the 'Info' icon (a blue circle with an 'i') next to the 'Subaccount' field in the 'Original Subaccount' section.

## Find Your Subaccount ID (Cloud Foundry Environment)

Get your subaccount ID to configure the Cloud Connector in the Cloud Foundry environment.

### **i 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:

The screenshot shows the SAP BTP Cockpit interface. The top navigation bar says 'SAP BTP Cockpit'. Below it, there is a table with three rows, each representing a subaccount. The first row is 'TEST\_CF\_AWS\_US' with provider 'Amazon Web Services (AWS)', region 'US East (VA) - Staging', description 'test', and environment 'Multi-Environment'. The second row is 'TEST\_CF\_AWS\_US\_2' with similar details. The third row is 'TEST\_CF' with provider 'Amazon Web Services (AWS)', region 'Europe (Frankfurt)', description '-none-', and environment 'Multi-Environment'. Each row has edit and info icons. A red box highlights the info icon in the 'TEST\_CF' row.

# 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.

Custom Regions (2)		Actions
<Custom Region 1>	<Region Host 1>	
<Custom Region 2>	<Region Host 2>	

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.

# Authenticating Users against On-Premise Systems

Authentication types supported by the Cloud Connector.

Currently, the Cloud Connector supports **basic authentication** and **principal propagation** (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](#).

- 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**, you must explicitly configure trust to those cloud entities from which user tokens are accepted as valid. You can do this in the **Trust** view of the Cloud Connector, see [Set Up Trust for Principal Propagation](#).

## Related Information

[Configuring Principal Propagation](#)

# Configuring Principal Propagation

Use principal propagation to simplify the access of SAP BTP users to on-premise systems.

Tasks in this section:

Task	Description
<a href="#">Set Up Trust for Principal Propagation</a>	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.
<a href="#">Configure a CA Certificate for Principal Propagation</a>	Install and configure an X.509 certificate to enable support for principal propagation.
<a href="#">Configuring Principal Propagation to an ABAP System</a>	Learn more about the different types of configuring and supporting principal propagation for a particular AS ABAP.
<a href="#">Configure Subject Patterns for Principal Propagation</a>	Define a pattern identifying the user for the subject of the generated short-lived X.509 certificate, as well as its validity period.
<a href="#">Configure a Secure Login Server</a>	Configuration steps for Java Secure Login Server (SLS) support.
<a href="#">Configure Kerberos</a>	The Cloud Connector lets you propagate users authenticated in SAP BTP via Kerberos against back-end systems. It uses the <b>Service For User and Constrained Delegation</b> protocol extension of Kerberos.
<a href="#">Configuring Principal Propagation to SAP NetWeaver AS for Java</a>	Find step-by-step instructions on how to configure principal propagation to an application server Java (AS Java).

## Related Information

[Principal Propagation](#) (Cloud Foundry environment)

[Principal Propagation](#) (Neo environment)

## Set Up Trust for Principal Propagation

Establish trust to an identity provider to support principal propagation.

### Tasks

[Configure Trusted Entities in the Cloud Connector](#)

[Configure an On-Premise System for Principal Propagation](#)

[Trust Cloud Applications in the Cloud Connector](#)

[Set up a Trust Store](#)

## Configure Trusted Entities in the Cloud Connector

You perform trust configuration to support principal propagation. 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.

### i 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, the title "Cloud Connector Administration", and a user "Administrator". The left sidebar has a "Connector" section with sub-options: Security Status, Alerting, High Availability, Hardware Metrics Monitor, and Configuration. Under "Configuration", there is a "Subaccounts" section with "conn3" selected. Below "conn3" are options: "Cloud To On-Premise" (which is highlighted in blue), "On-Premise To Cloud", "Monitor", "Audits", and "Log And Trace Files". The main content area is titled "Cloud To On-Premise" and has tabs: ACCESS CONTROL, COOKIE DOMAINS, APPLICATIONS, and PRINCIPAL PROPAGATION (which is underlined). The "PRINCIPAL PROPAGATION" tab displays a table titled "Trust Configuration (0)". The table has columns: Name, Description, Type, Trusted, and Actions. A note below the table says "Synchronize to obtain a list of IdPs or applications". There are icons for edit, delete, and synchronize in the header of the table.

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.

### i 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](#)

## 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.

### i 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:

- a. First, you must allow the Cloud Connector to identify itself with its system certificate (for HTTPS), or with the SNC PSE (for RFC).
- b. 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 \(HTTP\)](#).
  - For RFC, the Cloud Connector forwards the true identity as part of the RFC protocol.

For more information, see [Configuring Principal Propagation to an ABAP System](#).

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.

### i 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](#)

## Trust Cloud Applications in the Cloud Connector

Configure an allowlist for trusted cloud applications, see [Configure Trust](#).

Back to [Tasks](#)

## Set up a Trust Store

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

Configure a trust store that acts as an allowlist for trusted on-premise systems. See [Configure Trust](#).

Back to [Tasks](#)

## Related Information

[Principal Propagation](#) (Cloud Foundry)

[Principal Propagation](#) (Neo)

# Configure a CA Certificate for Principal Propagation

Install and configure an X.509 certificate to enable support for principal propagation in the Cloud Connector.

## Supported CA Mechanisms

You can enable support for principal propagation with X.509 certificates by performing either of the following procedures:

- Using a *Local CA* in the Cloud Connector.

**i 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](#).
- Option 3: (As of version 2.10) 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**.

**i Note**

The CA certificate should have the KeyUsage attribute `keyCertSign`. Many systems verify that the issuer of a certificate includes this attribute and deny a client certificate without this attribute. When using the CSR procedure, the attribute is

requested for the CA certificate. Also, when generating a self-signed certificate, this attribute is 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.

## ⚠ 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.

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](#).

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar has sections like Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, and Configuration (which is selected). The main area has tabs for USER INTERFACE, CLOUD, ON PREMISE (which is active), REPORTING, and ADVANCED. Under ON PREMISE, there's a 'CA Certificate' section. It shows 'Certificate Type: Local CA', 'Subject DN:', and 'Issuer:'. To the right is a table for 'Subject Alternative Names' with a single row: 'Type Value' and 'No data'. A toolbar above the table includes icons for refresh, save, upload, delete, and help.

A wizard offers in a first step a quick configuration by metadata URL pointing to the SLS you'd like to use.

The screenshot shows a configuration wizard step titled "Configure CA Via Secure Login Server". It contains a note: "You may leave this field empty and go to the next step to set up the configuration without metadata URL". Below is a "Meta URL:" label with an input field. At the bottom are "Previous", "Next" (highlighted in blue), and "Cancel" buttons.

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

**i 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.

**i 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.

**i 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.

**i 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:*	<input type="text" value="3"/>
Host Name:*	<input type="text" value="....."/>
Port:	<input type="text" value="443"/>

[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:*	<input type="text" value="....."/>
Authentication Port:	<input type="text" value="10443"/>

[Previous](#)[Finish](#)[Cancel](#)

Choose **Finish** to save the configuration.

## Related Information

[Configure a Secure Login Server](#)

[Initial Configuration \(HTTP\)](#)

[Initial Configuration \(RFC\)](#)

## Configuring Principal Propagation to an ABAP System

Learn more about the different types of configuring and supporting principal propagation for a particular AS ABAP.

Task	Description
<a href="#">Configure Principal Propagation for HTTPS</a>	Step-by-step instructions to configure principal propagation to an ABAP server for HTTPS.

Task	Description
<a href="#">Configure Principal Propagation via SAP Web Dispatcher</a>	Set up a trust chain to use principal propagation to an ABAP server for HTTPS via SAP Web Dispatcher.
<a href="#">Configure Principal Propagation for RFC</a>	Step-by-step instructions to configure principal propagation to an ABAP server for RFC.
<a href="#">Rule-Based Mapping of Certificates</a>	Map short-lived certificates to users in the ABAP server.

## Configure Principal Propagation for HTTPS

Find step-by-step instructions to configure principal propagation to an ABAP server for HTTPS.

### 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](#)

#### [Configure an ABAP System to Trust the Cloud Connector's System Certificate](#)

#### [Map Short-Lived Certificates to Users](#)

#### [Access ICF Services](#)

### Prerequisites

- For principal 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 \(HTTP\)](#)) must have specified the respective principal type to forward the principal 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](#)

Back to [Example Data](#)

## 1. Configure an ABAP System to Trust the Cloud Connector's System Certificate

This step includes two sub-steps:

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

[Configure the ABAP system to trust the Cloud Connector's system certificate](#)[Configure the Internet Communication Manager \(ICM\) to trust the system certificate for principal propagation](#)**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](#)

**Configure the Internet Communication Manager (ICM) to trust the system certificate for principal 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"**.

**i 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**

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

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 Principal Propagation via SAP Web Dispatcher](#).

### Caution

When using principal 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](#)

Back to [Tasks](#)

Back to [Example Data](#)

## 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](#).

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](#)

Back to [Example Data](#)

## 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 \(HTTP\)](#), 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 principal 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 \(HTTP\)](#), 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:
  - 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.
  - In the ABAP system, choose transaction code SICF and go to [Maintain Services](#).
  - Select the service that requires **Basic Authentication** as logon method.
  - Double-click the service and go to tab **Logon Data**.
  - 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 \(HTTP\)](#), 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 \(HTTP\)](#), 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:
  - In the Cloud Connector system mapping, choose the principal type **X.509 Certificate (general usage)** as described above.
  - In the ABAP system, choose transaction code SICF and go to [Maintain Services](#).
  - 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**.

### i 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](#)

Back to [Example Data](#)

## Related Information

[Configure Principal Propagation via SAP Web Dispatcher](#)

[Rule-Based Mapping of Certificates](#)

[Configure Subject Patterns for Principal Propagation](#)

[Set Up Trust for Principal Propagation](#)

[Principal Propagation](#) (Cloud Foundry environment)

[Principal Propagation](#) (Neo environment)

## Configure Principal Propagation via SAP Web Dispatcher

Set up a trust chain to use principal propagation to an ABAP System for HTTPS via SAP Web Dispatcher.

### 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 Principal Propagation for HTTPS](#)), 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](#)

- [Configure the SAP Web Dispatcher to Trust the Cloud Connector's System Certificate](#)
- [Next Steps](#)

## 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 SSL 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](#)

Back to [Concept](#)

## 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:

- a. To import the system certificate to SAP Web Dispatcher, open the SAP Web Dispatcher administration interface in your browser.

#### **i Note**

The interface is usually configured on `/sap/wdisp/admin`.

- b. In the menu, navigate to **SSL and Trust Configuration** and select **PSE Management**.
- c. 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.
- d. In the **Trusted Certificates** section, choose **Import Certificate**.
- e. 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.

#### **i Note**

Typically, this is a CA certificate. If you are using a self-signed system certificate, it's the system certificate itself.

- f. Choose **Import**.

- g. The certificate details are now shown in section **Trusted Certificates**.

### 2. Configure SAP Web Dispatcher to trust the Cloud Connector's system certificate for principal propagation:

- o 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"`

- [Deprecated] Create or edit the following two parameters in SAP Web Dispatcher:

#### i 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 )

#### i 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](#)

Back to [Concept](#)

## Next Steps

Now you can proceed with:

- Step 1 of the basic principal propagation setup for HTTPS, see [Configure an ABAP System to Trust the Cloud Connector's System Certificate](#). 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 principal propagation setup for HTTPS as described here:

- [Map Short-Lived Certificates to Users](#)
- [Access ICF Services](#)

Back to [Tasks](#)

Back to [Concept](#)

## Configure Principal 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.

### i 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:

### i 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.

### i 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](#)
2. [Map Short-Lived Certificates to Users](#)
3. [Configure the Cloud Connector](#)

## 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](#)[Back to Example Data](#)

## 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](#)[Back to Example Data](#)

## 3. Configure the Cloud Connector

[Prerequisites](#)[Set up the Cloud Connector to Use the SNC Implementation](#)[Create an RFC Hostname Mapping](#)**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.

### **i Note**

If you use SAP Cryptographic Library as SNC implementation, follow the steps described in [Initial Configuration \(RFC\)](#). 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](#)[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 \(RFC\)](#).

### **i Note**

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

The example in [Initial Configuration \(RFC\)](#) 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](#)

## 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 \(RFC\)](#).
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 identitiy name, for example, p:CN=SID, O=Trust Community, C=DE.
3. Save your mapping.

Back to [Step](#)

Back to [Tasks](#)

Back to [Example Data](#)

## Related Information

[Secure Network Communications \(SNC\)](#)

[Using the SAP Cryptographic Library for SNC](#)

[Rule-Based Mapping of Certificates](#)

[Principal Propagation](#) (Cloud Foundry environment)

[Principal Propagation](#) (Neo environment)

## Rule-Based Mapping of Certificates

Learn how to efficiently map short-lived certificates to users in the ABAP server.

To perform rule-based mapping of certificates in the ABAP server, proceed as follows:

1. Enter a dynamic parameter using transaction RZ11.
  - a. Enter the `login/certificate_mapping_rulebased` parameter.
  - b. Choose the **Change Value** button.
  - c. Enter the value 1.
  - d. Save the value.

**i Note**

If dynamic parameters are disabled, enter the value using transaction RZ10 and restart the whole ABAP system.

## 2. Configure rule-based mapping

- a. 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.
- b. To import the sample certificate into the ABAP server, choose transaction CERTRULE and select **Import certificate**.

**i Note**

To access transaction CERTRULE, you need the corresponding authorizations (see: [Assign Authorization Objects for Rule-based Mapping](#)).

- c. To create explicit rule mappings, choose the **Rule** button.
- d. Choose **Save**.

**i 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](#)

## 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**.

## Configure Subject Patterns for Principal Propagation

Define patterns identifying the user for the subject of a generated short-lived X.509 certificate.

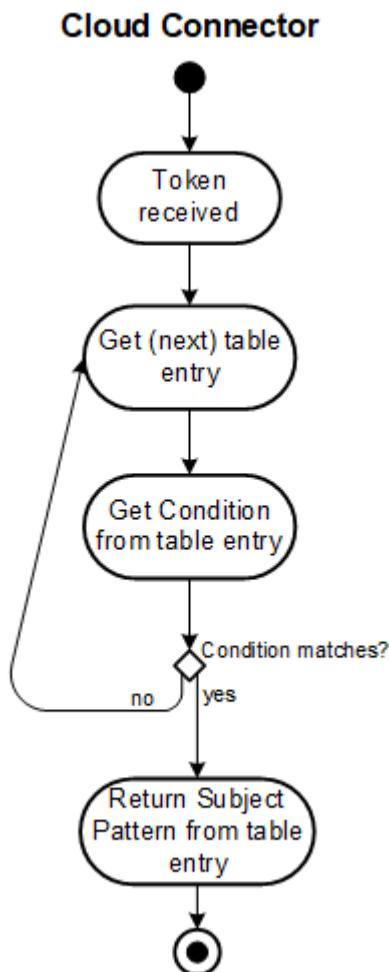
Using this configuration option, you can define different patterns identifying the user for the subject of the generated short-lived X.509 certificate, based on a specified condition. You can also specify the validity period and expiration tolerance.

### Configure Subject Patterns

To configure a subject pattern, choose **Configuration > On Premise > Principal Propagation**. In the table shown, you can add or modify patterns.

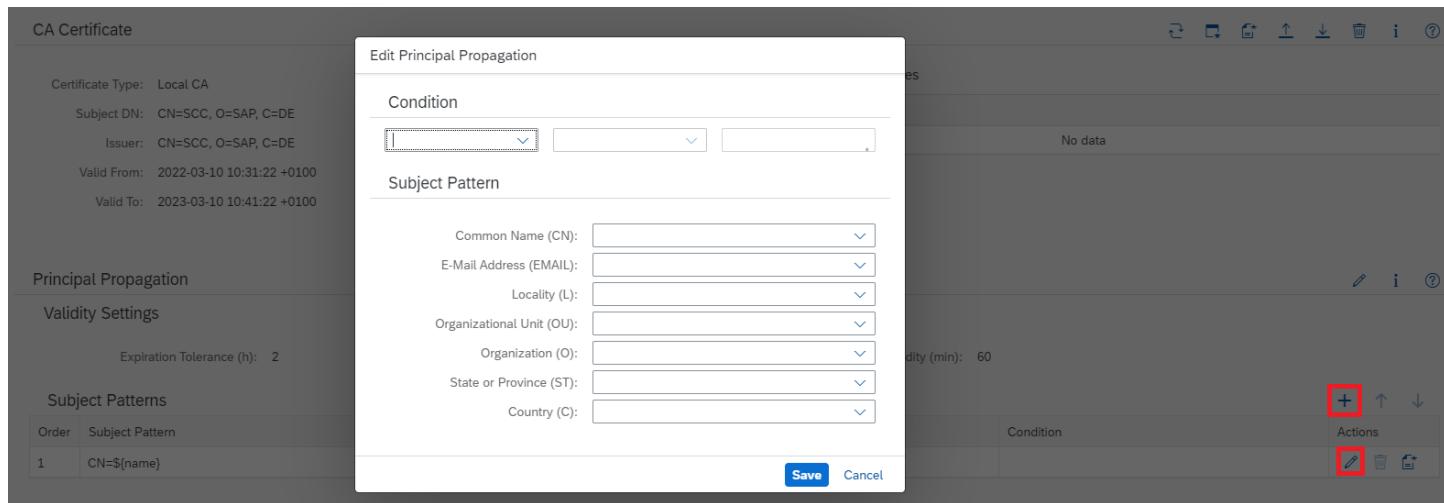
Subject Patterns			<b>Actions</b>
Order	Subject Pattern	Condition	
1	CN=\${mail}	\${mail} exists	
2	CN=\${name}		

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.

### **i 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}

### i 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.

**When using a subaccount in the Cloud Foundry environment:** As of version 2.12.5, 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.

The screenshot shows a SAP Fiori application window titled 'Principal Propagation'. On the left, there's a table for 'Subject Patterns' with two rows: '1 CN=\${mail}' and '2 CN=\${name}'. A modal dialog box titled 'Create Sample Certificate' is open in the center. It contains a field 'CN mail:' with the value 'mail'. At the bottom of the dialog are 'Generate' and 'Cancel' buttons. In the top right corner of the dialog, there's a red box highlighting the 'Save' button. The background of the main screen shows a table with a condition '\${mail} exists' and actions for edit, delete, and copy.

## Validity Settings

You can change the validity settings by choosing the **Edit** button.

The screenshot shows a SAP Fiori application window titled 'Principal Propagation'. On the left, there's a table for 'Subject Patterns' with one row: '1 CN=\${mail}'. A modal dialog box titled 'Edit Principal Propagation' is open in the center. It contains fields for 'Expiration Tolerance (h):' (value '2') and 'Certificate Validity (min):' (value '60'). At the bottom of the dialog are 'Save' and 'Cancel' buttons. In the top right corner of the dialog, there's a red box highlighting the 'Save' button. The background of the main screen shows a table with a condition '\${mail} exists' and actions for edit, delete, and copy.

## Related Information

[Server Certificate Authentication](#)

## Configure a Secure Login Server

Configuration steps for Java SLS support.

## Content

[Overview](#)

[Requirements](#)

[Implementation](#)

## 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.

### i 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

### **i 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](#)

## 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](#)

## Implementation

### INSTALLATION

Follow the standard installation procedures for SLS. This includes the initial setup of a PKI (public key infrastructure).

### **i 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".

## i 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**.

## i 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:
  - a. full DNS name
  - b. 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:
  - a. Full DNS name
  - b. 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:

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](#)

## Configure Kerberos

### Context

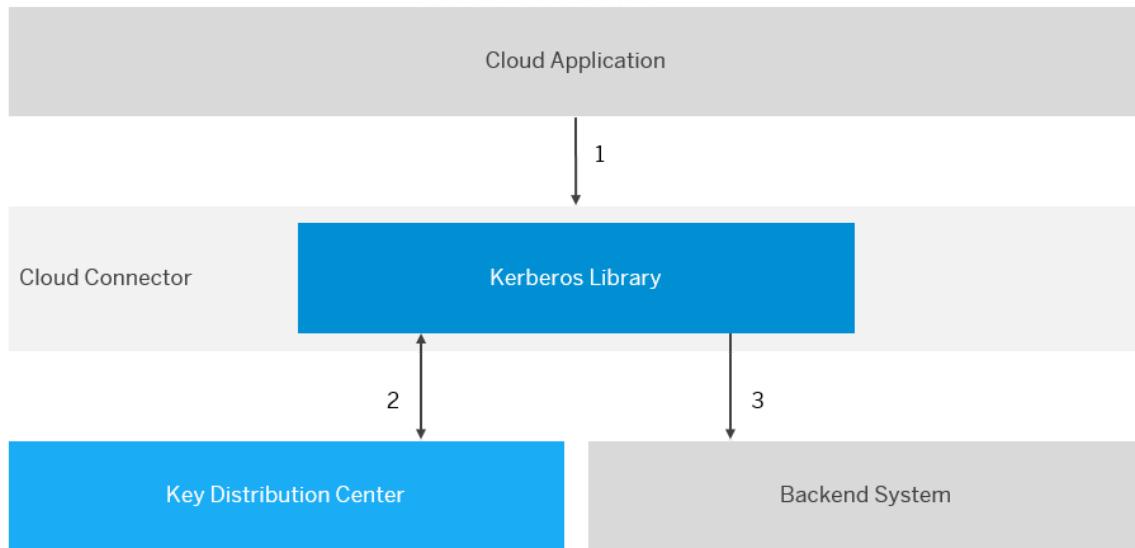
The Cloud Connector allows you to propagate users authenticated in SAP BTP via Kerberos against backend systems. It uses the **Service For User and Constrained Delegation** protocol extension of Kerberos.

#### **i** 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 for Principal Propagation](#).

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**.

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a navigation sidebar with options like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. Under 'Configuration', a subaccount 'conn2' is selected, with 'Cloud To On-Premise' and 'On-Premise To Cloud' listed under it. The main area is titled 'Configuration' and has tabs for 'USER INTERFACE', 'CLOUD', and 'ON PREMISE'. The 'ON PREMISE' tab is active, showing 'Subject Pattern: CN=\${name}'. Below this, there are fields for 'Expiration Tolerance (h): 2' and 'Certificates Validity (min): 60'. A modal window titled 'Edit Kerberos' is open over the main interface. This modal has two sections: 'KEYTAB' and 'KDC Hosts'. In the 'KEYTAB' section, there are three input fields: 'Realm Name:' with a placeholder 'realmname', 'KEYTAB File:' with a placeholder 'keytabfile' and a 'Browse' button, and 'User Name:' with a placeholder 'username'. In the 'KDC Hosts' section, there's a table with columns 'Host Name', 'Port', and 'Actions'. The table currently shows 'No data'. At the bottom of the modal are 'Save', 'Cancel', and 'Help' buttons.

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 for Principal Propagation](#)

## Configuring Principal 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 principal propagation for HTTPS.

## 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 principal 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 principal 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](#).
  - o To map the user ID of the certificate's subject name field to users, see [Using Rules Based on Client Certificate Subject Names](#).
  - o To map the user ID based on rules for the certificate V3 extension *SubjectAlternativeName*, see [Using Rules Based on Client Certificate V3 Extensions](#).
  - o 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 Principal Propagation for HTTPS](#)

## 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 \(HTTP\)](#)

[Configure Access Control \(RFC\)](#)

[Configure Access Control \(LDAP\)](#)

[Configure Access Control \(TCP\)](#)

## 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

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar has a 'Connector' section with 'Cloud To On-Premise' selected. The main area shows the 'Cloud To On-Premise' configuration. In the center, there's a modal dialog titled 'Import System Mappings'. It contains fields for 'Source': 'File' (radio button selected) or 'Subaccount' (radio button unselected), a 'File:' input field with a 'Browse' button, an 'Overwrite:' checkbox (unchecked), and an 'Include Resources:' checkbox (checked). At the bottom are 'Import' and 'Cancel' buttons.

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).

### i 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 \(HTTP\)](#)

[Configure Access Control \(RFC\)](#)

[Configure Access Control \(LDAP\)](#)

[Configure Access Control \(TCP\)](#)

[Configure Accessible Resources](#)

[Configure Domain Mappings for Cookies](#)

## 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.

### i 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](#)

[Limit the Accessible Services for HTTP\(S\)](#)

[Activate or Suspend Resources](#)

## 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**.

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a sidebar with options like Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, and a trial subaccount under Cloud To On-Premise. The main area is titled 'Cloud To On-Premise' and has tabs for ACCESS CONTROL, COOKIE DOMAINS, APPLICATIONS, and PRINCIPAL PROPAGATION. A sub-section titled 'Mapping Virtual To Internal System (0)' is selected. A modal dialog box is open, titled 'Add System Mapping'. Inside the dialog, there's a step counter '0 / 1' with a help icon, a dropdown menu for 'Select back-end type of on-premise system' currently set to 'ABAP System', and navigation buttons 'Previous', 'Next', and 'Cancel'.

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 SSL tunnel) is SSL-encrypted. The only prerequisite is that the backend system supports HTTPS on that port. For more information, see [Initial Configuration \(HTTP\)](#).

- o 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.
- o If there is no system certificate imported, the Cloud Connector opens an HTTPS connection without client certificate.

The screenshot shows the SAP Cloud Connector Administration interface. The sidebar and main area are similar to the previous screenshot. The 'Add System Mapping' dialog is now at step 1 of 1. It has a help icon and a dropdown for 'Protocol' set to 'HTTPS'. Navigation buttons 'Previous', 'Next', and 'Cancel' are at the bottom.

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.

The screenshot shows the SAP Cloud Connector Administration interface. The sidebar and main area are similar to the previous screenshots. The 'Add System Mapping' dialog is now at step 2 of 2. It has a help icon and fields for 'Internal Host' containing 'www.test.com' and 'Internal Port' containing '1443'. Navigation buttons 'Previous', 'Next', and 'Cancel' are at the bottom.

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](#) (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.

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a sidebar with options like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. Under 'Configuration', the 'trial' subaccount is selected. In the main area, under 'Cloud To On-Premise', there's a 'Mapping Virtual Host' section. A modal window titled 'Add System Mapping' is open, prompting the user to enter a virtual host and port. The 'Virtual Host:' field contains 'testsys.cloud' and the 'Virtual Port:' field contains '443'. A note in the modal says, 'It is recommended to use a virtual (cloud-side) name that is different from internal name'. At the bottom of the modal are 'Previous', 'Next', and 'Cancel' buttons.

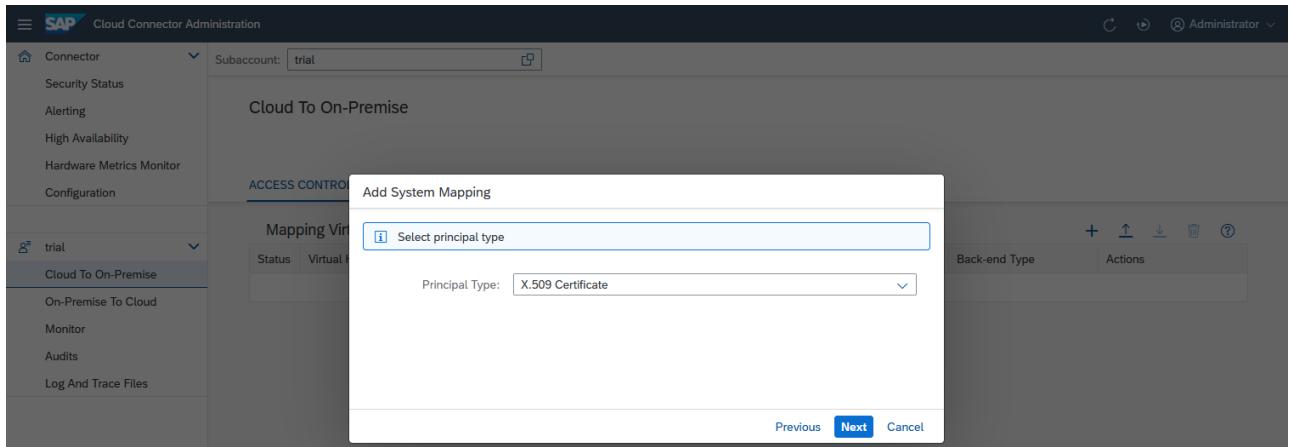
7. **Allow Principal Propagation** (available as of Cloud Connector 2.15): Defines if any kind of principal propagation should be allowed over this mapping. If not selected, go to step 9.

This screenshot shows the 'Add System Mapping' dialog. It includes a note: 'Choose whether Principal Propagation will be possible'. Below it is a checkbox labeled 'Allow Principal Propagation' which is checked. At the bottom are 'Previous', 'Next', and 'Cancel' buttons.

## 8. **Principal Type**

- o *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 backend configuration, it can be used for authentication.



- o *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.

### i 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](#).

**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

- a. No principal is received, or
- b. 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.

### i 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.

Add System Mapping

**Choose whether the System Certificate will be used for logon in case no Principal is received from Cloud**

System Certificate for Logon:

[Previous](#) [Next](#) [Cancel](#)

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.

Select host for request header field HOST

Host In Request 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**).

Optional enter a description

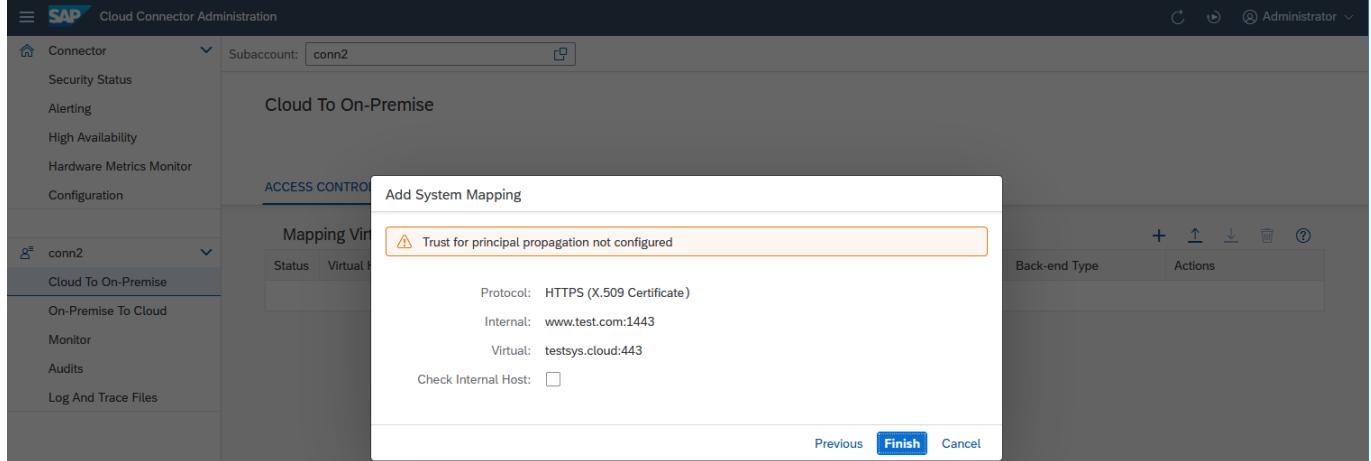
Description:

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.

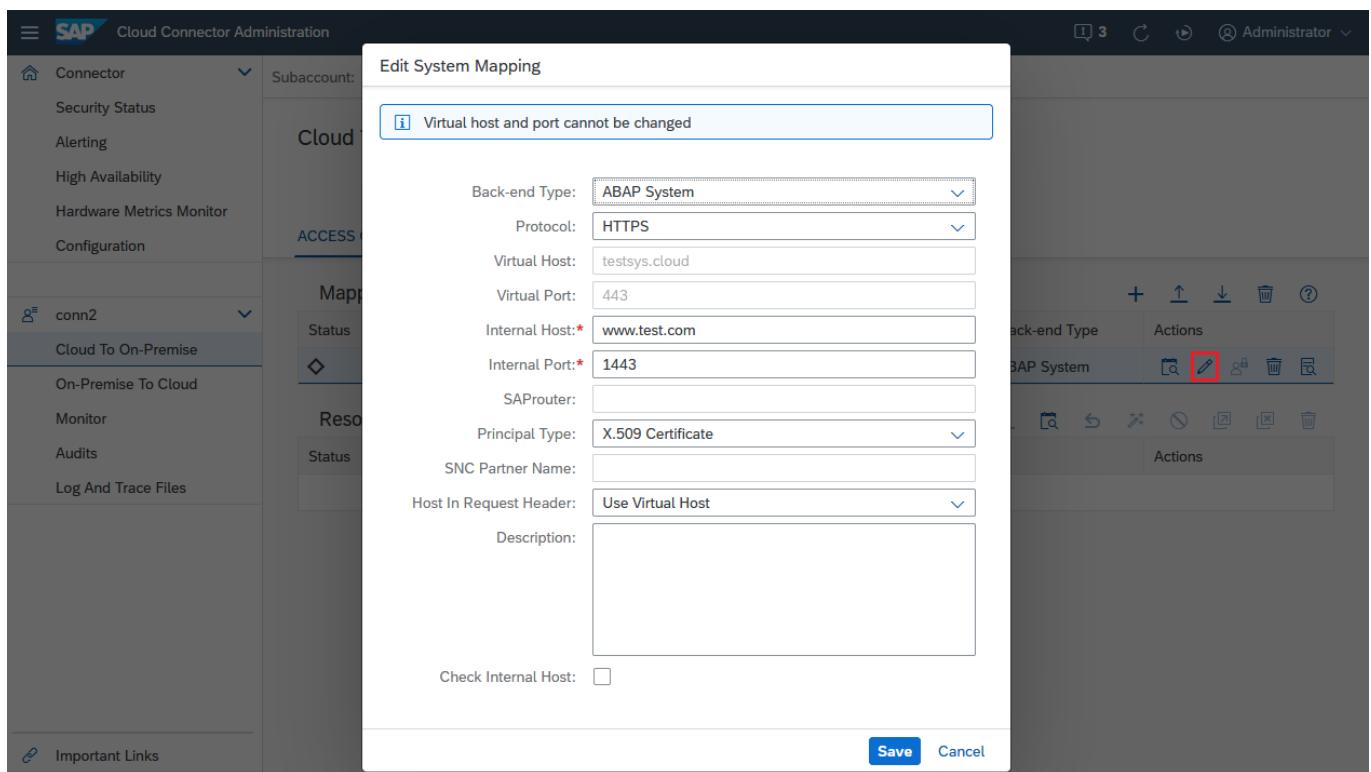
If the ping to the internal host is successful (that is, the host is reachable via TLS), the state **Reachable** is shown. If it fails, a warning pops up. You can view issue details by choosing the **Details** button, or check them in the log files.

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**.



13. 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](#)

## 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 screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a sidebar with options like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', 'Configuration', and a section for 'conn2' which includes 'Cloud To On-Premise' (selected), 'On-Premise To Cloud', 'Monitor', 'Audits', and 'Log And Trace Files'. The main area is titled 'Cloud To On-Premise' and shows a 'Mapping Virtual To...' table with one row: 'Status' and 'Virtual Host' both set to 'testsys.cloud:443'. Below this is a 'Resources Of testsys' table with columns 'Status' and 'URL Path'. A modal window titled 'Add Resource' is open, containing fields: 'URL Path:' with value '/public/test', 'Active:' checked, 'WebSocket:' unchecked, 'Access Policy:' with 'Path Only (Sub-Paths Are Excluded)' selected, and a 'Description:' field. At the bottom of the modal are 'Save' and 'Cancel' buttons.

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](#)

## 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.

Resources Of testsys.cloud:443 (1)			
Status	URL Path	Access Policy	Actions
<input checked="" type="checkbox"/>	/public/test	Path Only (Sub-Paths Are Excluded)	<a href="#"></a> <a href="#"></a> <a href="#"></a> <a href="#"></a>

- 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.

## Related Information

[Configure Domain Mappings for Cookies](#)

[Consume Backend Systems \(Java Web or Java EE 6 Web Profile\)](#)

# Configure Access Control (RFC)

Specify the backend systems that can be accessed by your cloud applications using RFC.

## Tasks

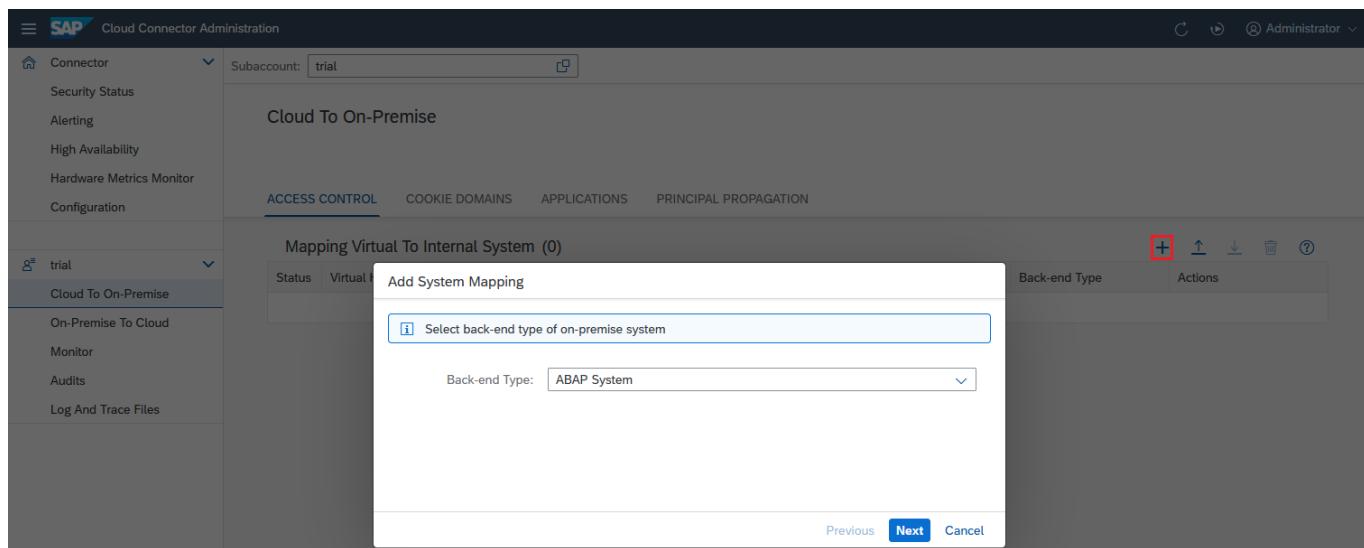
[Expose Intranet Systems](#)

[Limit the Accessible Resources for RFC](#)

## 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.

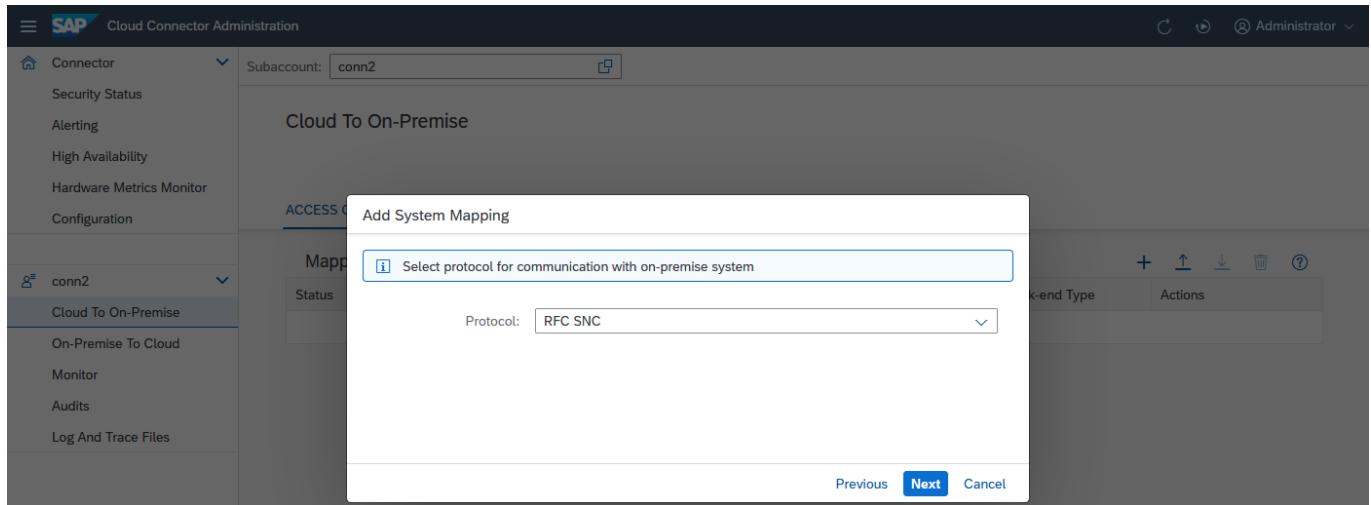
1. Choose **Cloud To On-Premise** from your **Subaccount** menu and go to tab **Access Control**.
2. Choose **Add**.
3. **Backend Type:** Select the backend system type ( ABAP System or SAP Gateway for RFC).



4. Choose **Next**.
5. **Protocol:** Choose RFC or RFC SNC for connecting to the backend system.

### i 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 \(RFC\)](#).

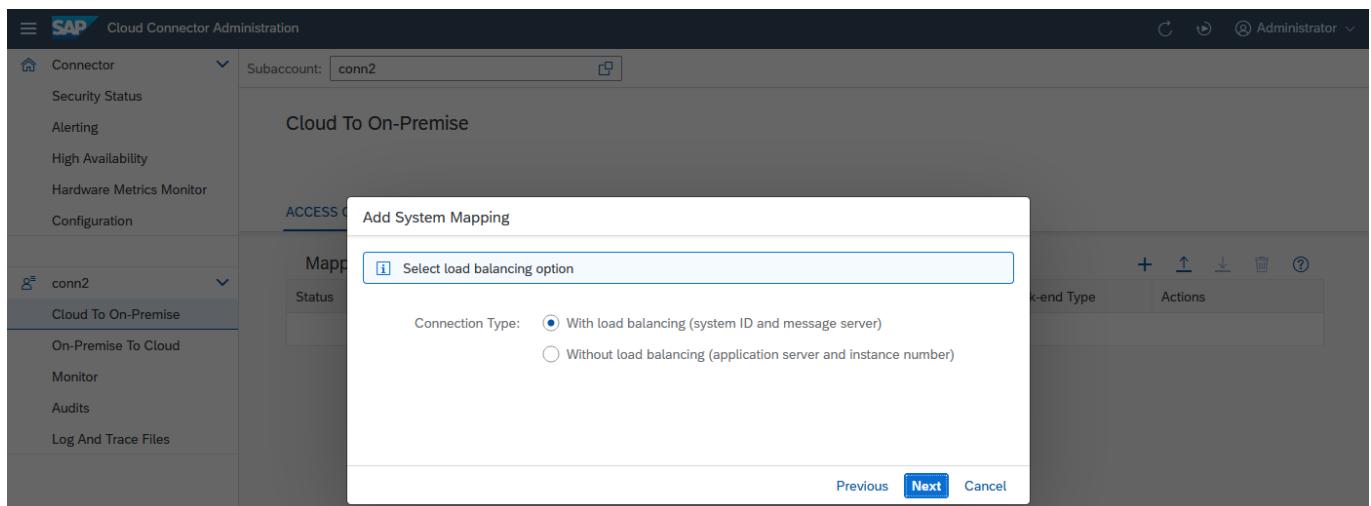


## i Note

- o The back end must be properly configured to support SNC connections.
- o SNC configuration must be provided in the Cloud Connector.

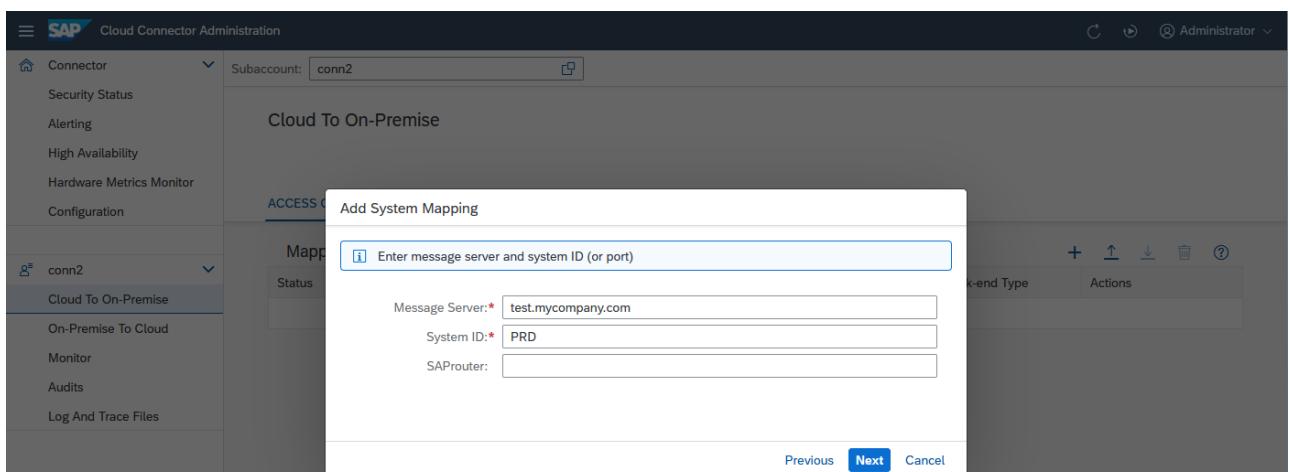
6. Choose **Next**.

7. Choose whether you want to configure a load balancing logon or connect to a specific application server.

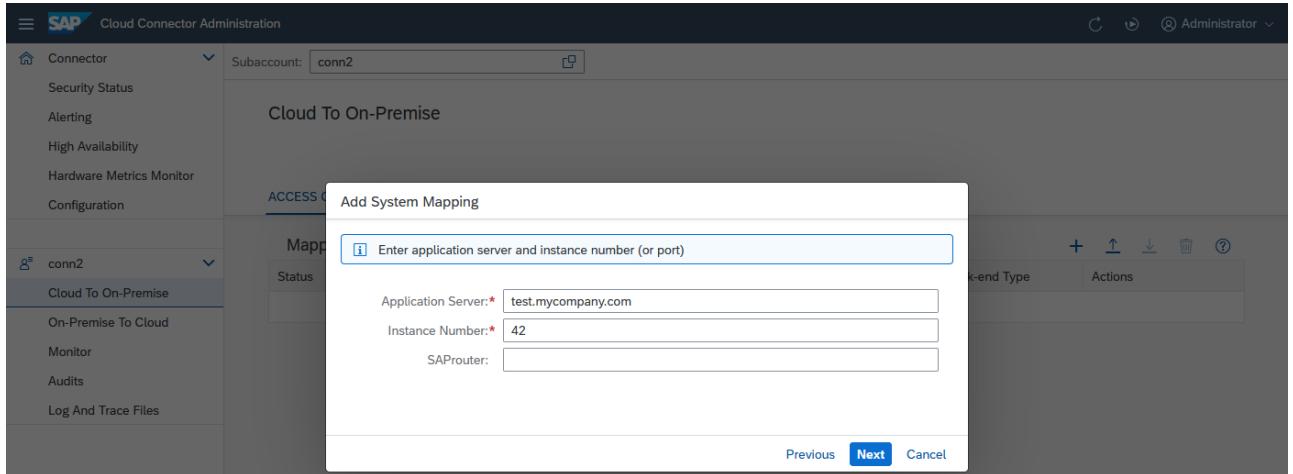


8. Specify the parameters of the backend system. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. If this is only possible using a valid [SAProuter](#), specify the router in the respective field. The Cloud Connector will try to establish a connection to this system, so the address has to be real.

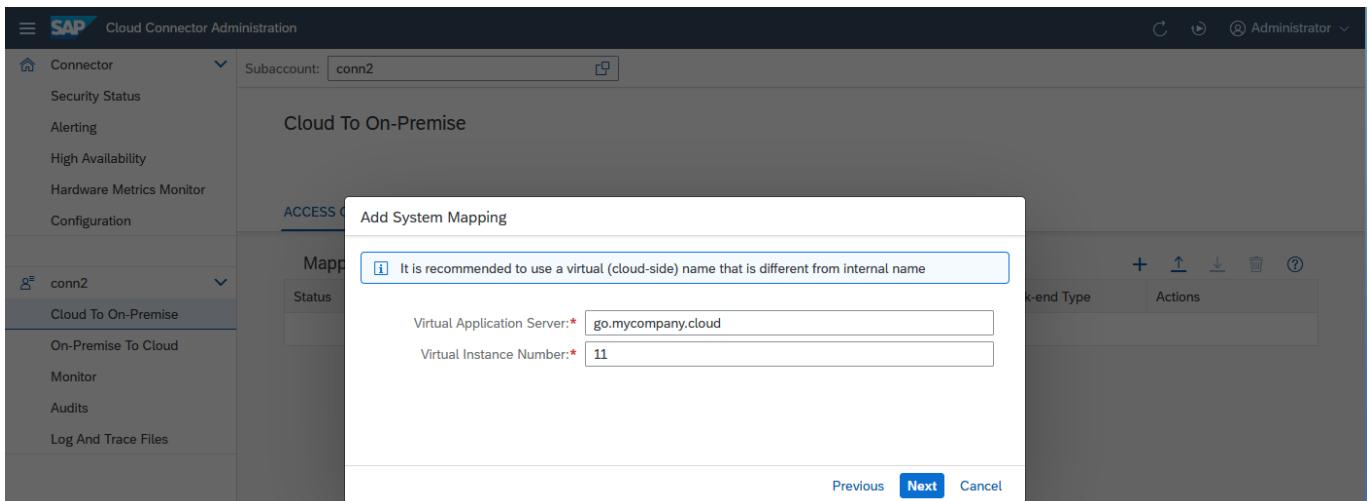
- o When using a load-balancing configuration, the **Message Server** specifies the message server of the ABAP system. The system ID is a three-char identifier that is also found in the SAP Logon configuration. Alternatively, it's possible to directly specify the message server port in the **System ID** 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**.



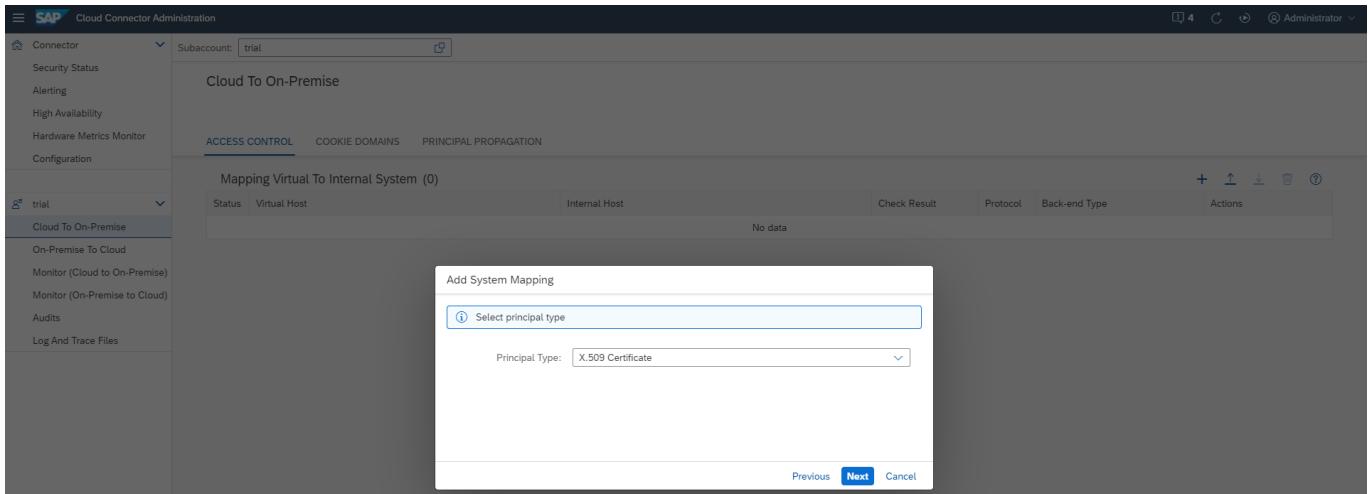
- Virtual Message Server** - specifies the host name exactly as specified as the `jco.client.mshost` property in the RFC destination configuration in the cloud. The **Virtual System ID** allows you to distinguish between different entry points of your backend system that have different sets of access 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.

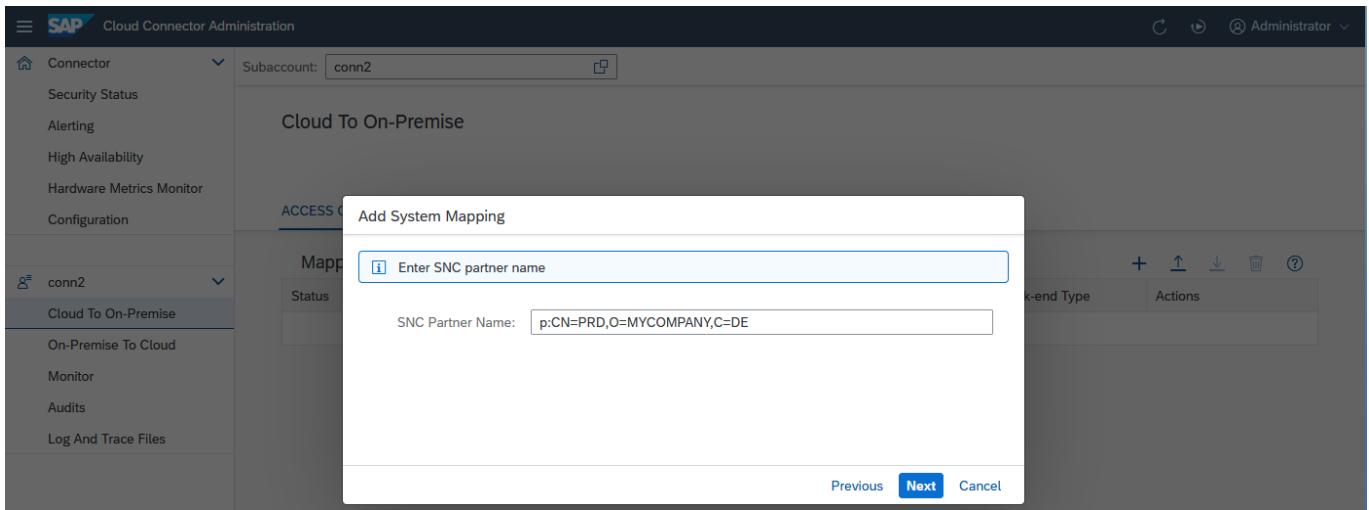
### i Note

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.

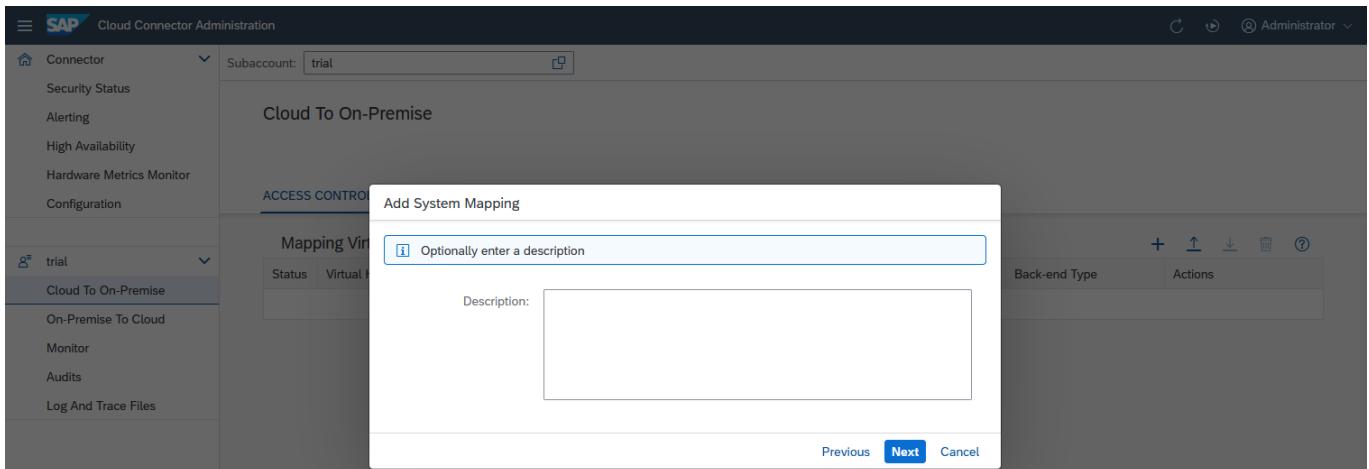
For more information on principal propagation, see [Configuring Principal Propagation](#).



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**.

## i Note

This function applies for RFC/RFC SNC only.

Protocol	Back-end Type	Actions
RFC	ABAP System	
RFC	ABAP System	
RFC SNC	ABAP System	
HTTPS	ABAP System	
HTTP	SAP Business Connector	
HTTPS	SAP Business Connector	
HTTPS	SAP Business Connector	
HTTP	SAP Business Connector	
HTTPS	ABAP System	

Back to [Tasks](#)

## 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.

Protocol	Back-end Type	Actions
ABAP System		
ABAP System		
ABAP System		
SAP Business Connector		
ABAP System		

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.
4. The **Active** checkbox allows you to specify whether that resource should be initially enabled or disabled.

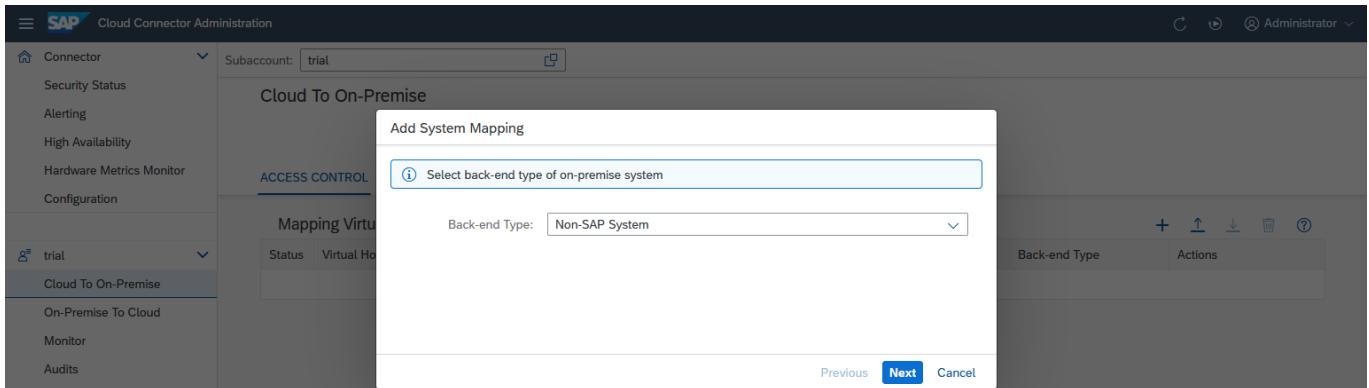
Back to [Tasks](#)

## Configure Access Control (LDAP)

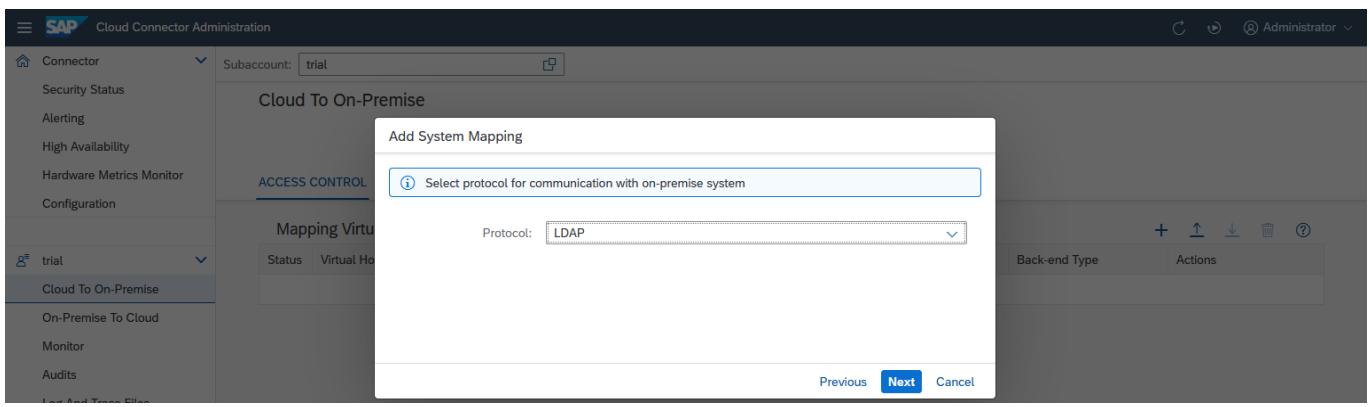
Add a specified system mapping to the Cloud Connector if you want to use an on-premise LDAP server or user authentication in your cloud application.

To allow your cloud applications to access an on-premise LDAP server, insert a new entry in the Cloud Connector access control management.

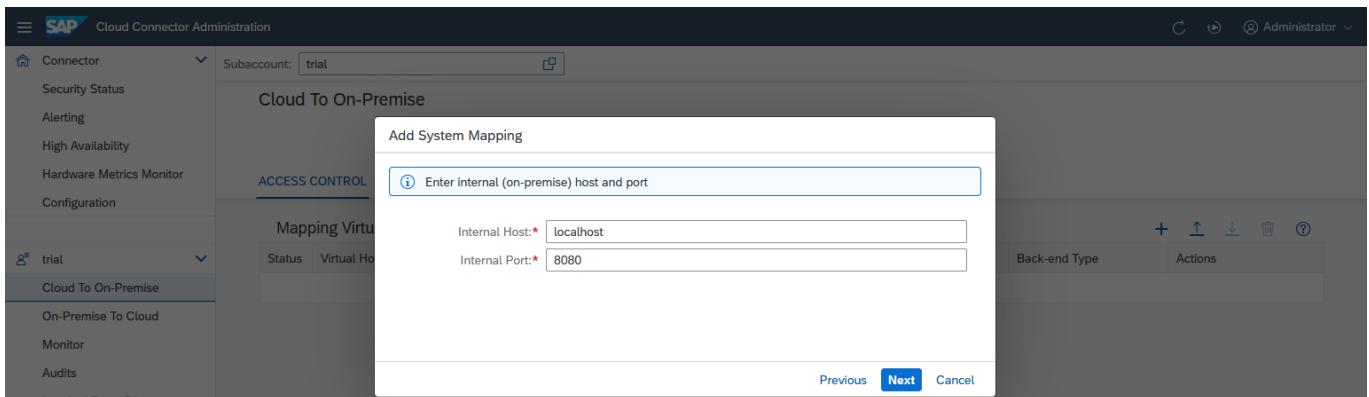
1. Choose **Cloud To On-Premise** from your **Subaccount** menu.
2. Choose **Add (+)**. A wizard opens and asks for the required values.
3. **Backend Type:** Select **Non-SAP System** from the drop down list. When you are done, choose **Next**.



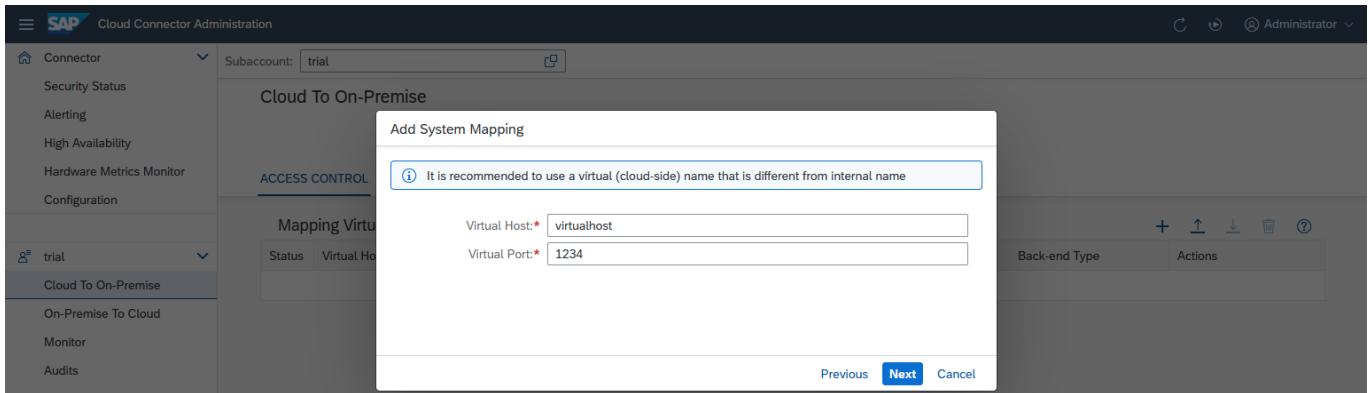
4. **Protocol:** Select LDAP or LDAPS for the connection to the backend system. When you are done, choose **Next**.



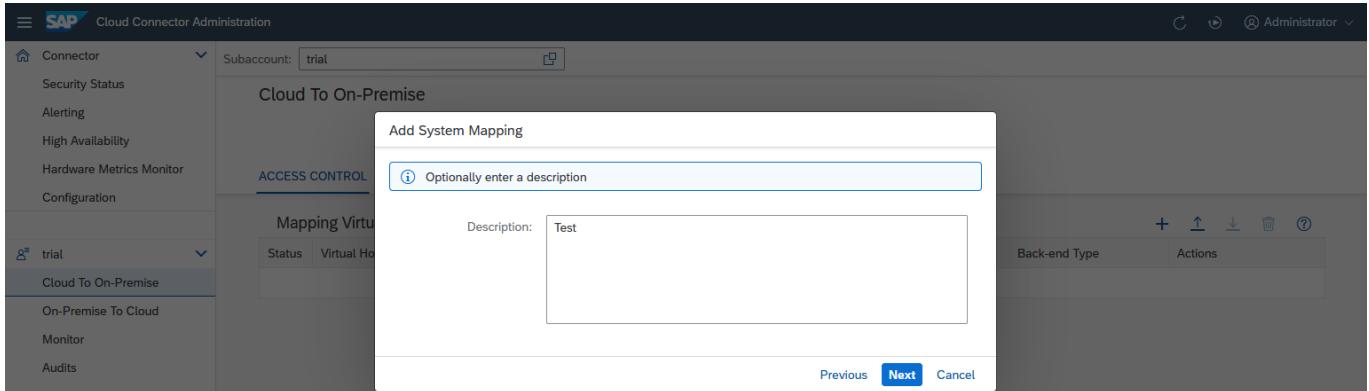
5. **Internal Host** and **Internal Port**: specify the host and port under which the target system can be reached within the intranet. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. The Cloud Connector will try to forward 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**.



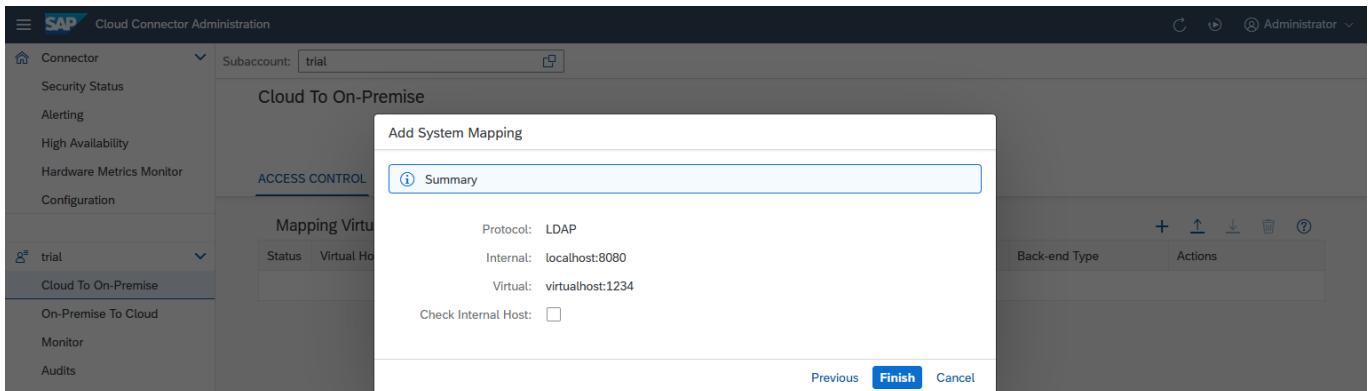
7. You can enter an optional description at this stage. The respective description will be shown as a tooltip when you press the button **Show Details** in column **Actions** of the **Mapping Virtual To Internal System** overview.



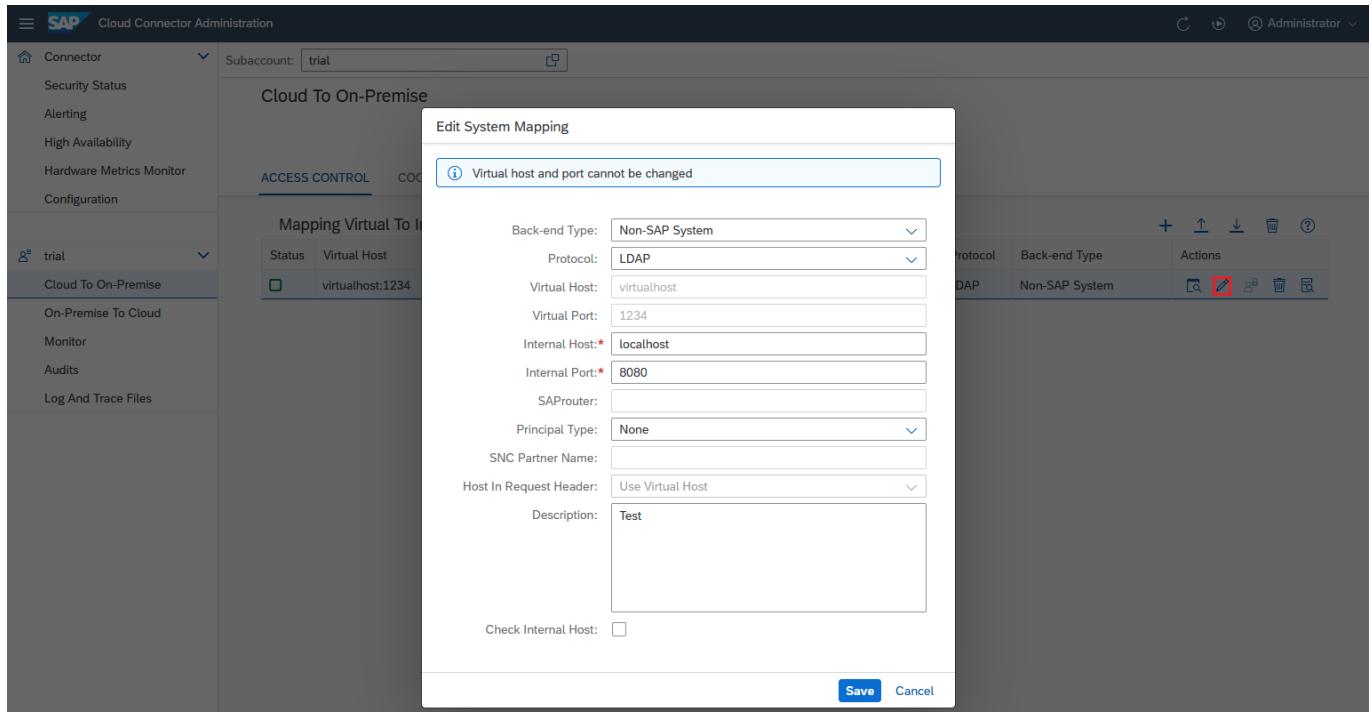
8. The summary shows information about the system to be stored. When saving the host mapping, you can trigger a ping from the Cloud Connector to the internal host, using the **Check Internal Host** check box. This allows you to make sure the Cloud Connector can indeed access the internal system. Also, you can catch basic things, such as spelling mistakes or firewall problems between Cloud Connector and 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**.



9. 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.

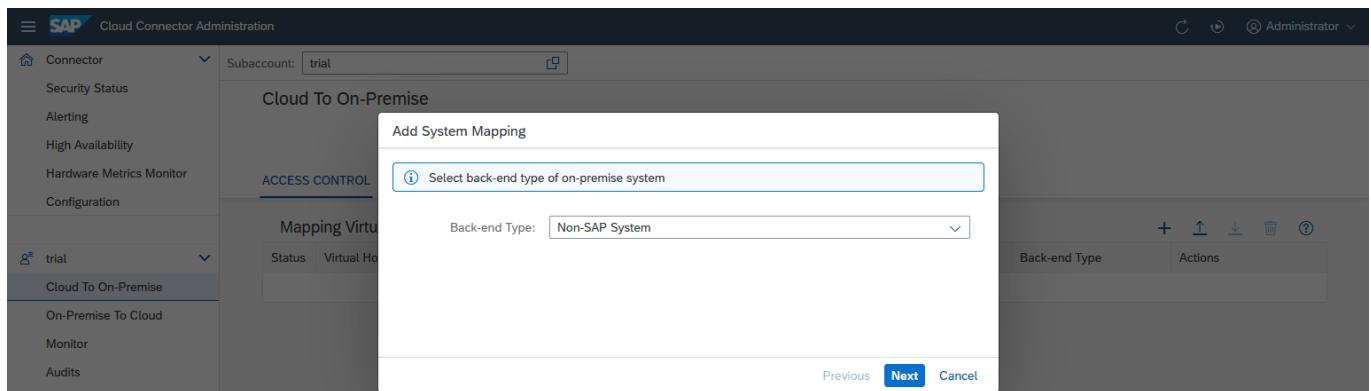


## Configure Access Control (TCP)

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.

1. Choose **Cloud To On-Premise** from your **Subaccount** menu.
2. Choose **Add (+)**. A wizard opens and asks for the required values.
3. **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**.

### i Note

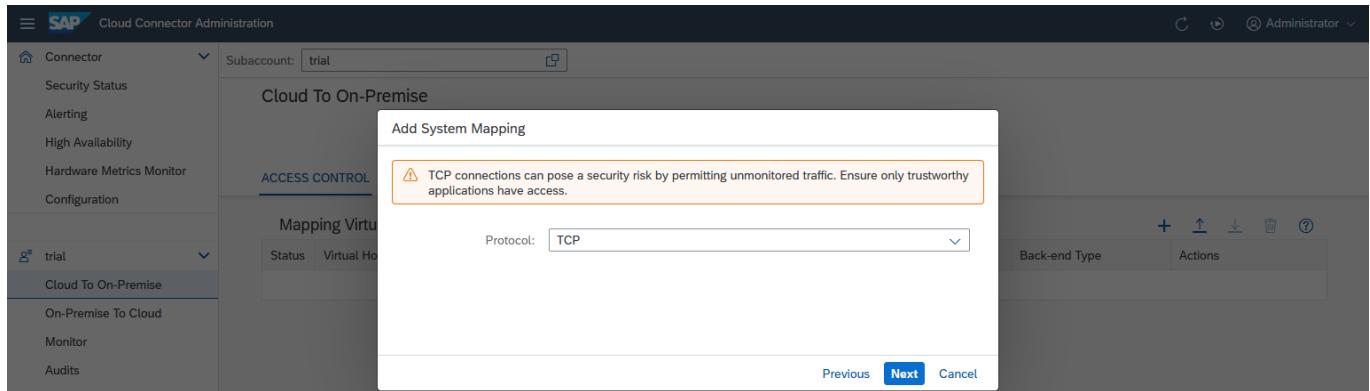
When selecting TCP as protocol, the following warning message is displayed:

TCP connections can pose a security risk by permitting unmonitored traffic. Ensure only trustworthy applications have access

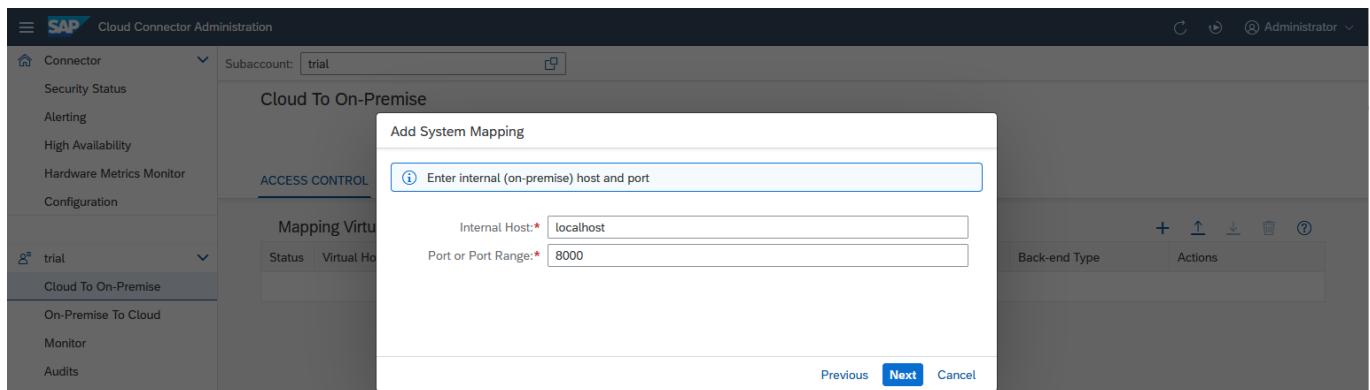
. The reason is that using plain TCP, the Cloud Connector cannot see or log any detail information about the calls.

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

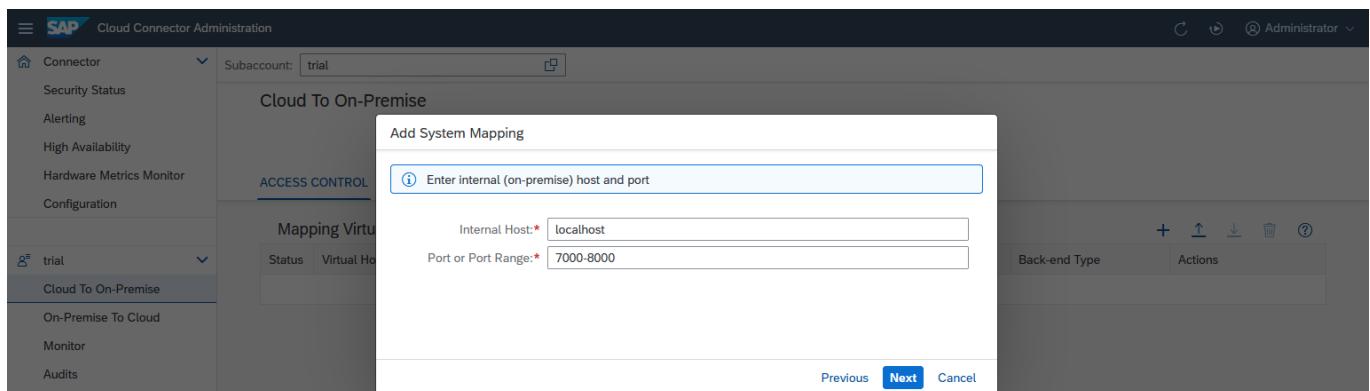
- deploy only trusted applications on SAP BTP.
- configure an application allowlist in the Cloud Connector, see [Set Up Trust for Principal Propagation](#).
- take the recommended security measures for your SAP BTP (sub)account. See section [Security](#) in the SAP BTP documentation.



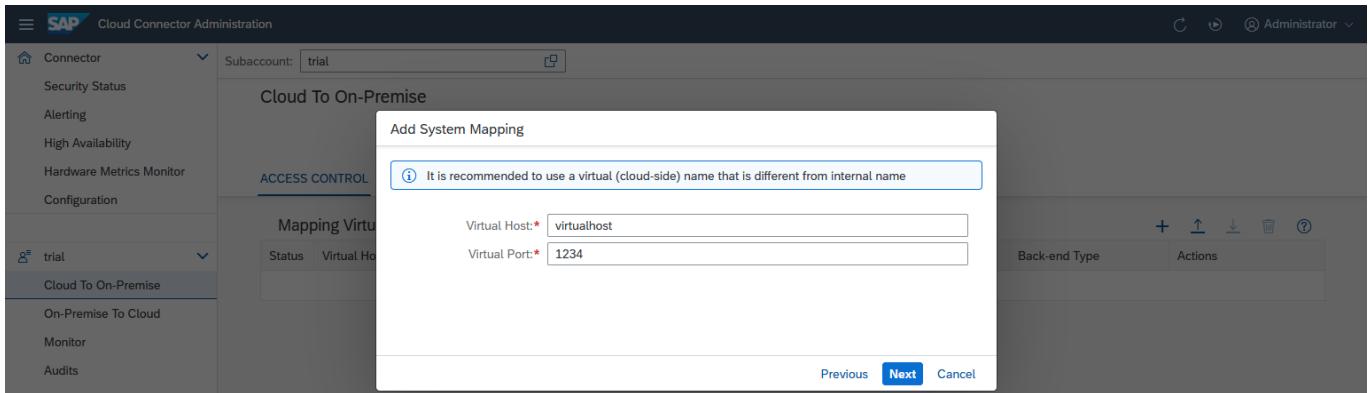
5. **Internal Host** and **Port or Port Range**: specify the host and port under which the target system can be reached within the intranet. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. The Cloud Connector will try to forward the request to the network address specified by the internal host and port. That is why this address needs to be real.



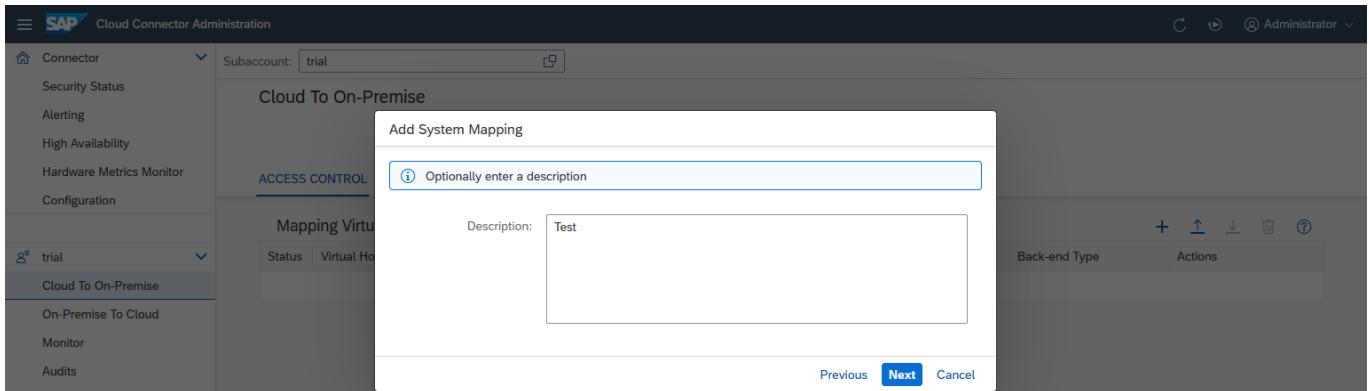
For TCP and TCP SSL, you can also specify a port range through its lower and upper limit, separated by a hyphen.



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.

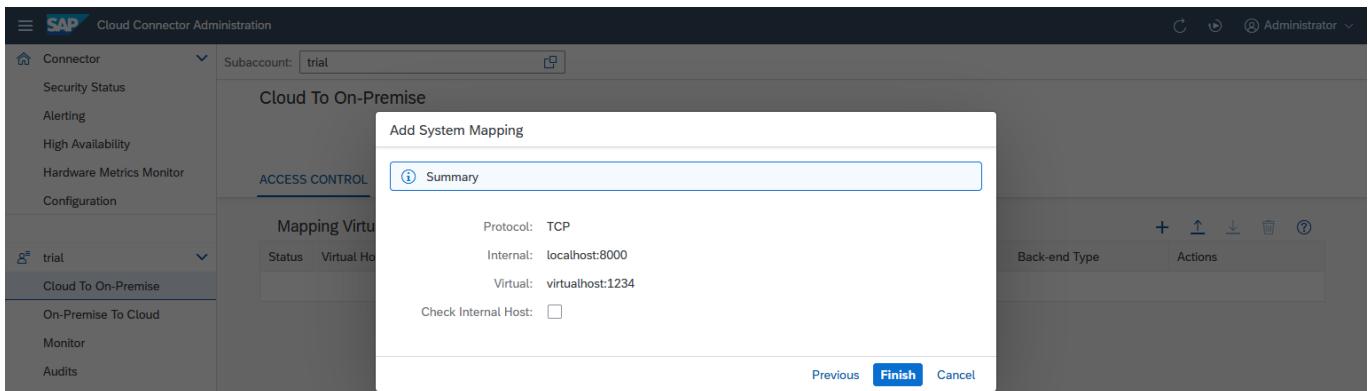


8. The summary shows information about the system to be stored. When saving the host mapping, you can trigger a ping from the Cloud Connector to the internal host, using the **Check Internal Host** checkbox. This allows you to make sure the Cloud Connector can access the internal system. Also, you can 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.

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar has a tree view with nodes like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', 'Configuration', 'trial' (selected), 'Cloud To On-Premise' (selected), 'On-Premise To Cloud', 'Monitor', 'Audits', and 'Log And Trace Files'. The main area has tabs for 'ACCESS CONTROL', 'COOKIE DOMAINS', 'APPLICATIONS', and 'PRINCIPAL PROPAGATION'. A modal window titled 'Edit System Mapping' is open, showing fields for 'Back-end Type' (Non-SAP System), 'Protocol' (TCP), 'Virtual Host' (virtualhost), 'Virtual Port' (1234), 'Internal Host' (localhost), 'Internal Port' (8000), 'SAProuter', 'Principal Type' (None), 'SNC Partner Name', 'Host In Request Header' (Use Virtual Host), and 'Description' (Test). A warning message at the top right of the modal says: 'TCP connections can pose a security risk by permitting unmonitored traffic. Ensure only trustworthy applications have access.' At the bottom of the modal are 'Save' and 'Cancel' buttons.

## 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](#)

[Use Scenarios for Resources](#)

### 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 has a tree view with nodes like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', 'Configuration', 'trial' (selected), 'Cloud To On-Premise' (selected), 'On-Premise To Cloud', 'Monitor', 'Audits', and 'Log And Trace Files'. The main area has tabs for 'ACCESS CONTROL', 'COOKIE DOMAINS', 'APPLICATIONS', and 'PRINCIPAL PROPAGATION'. A table titled 'Mapping Virtual To Internal System (7)' is displayed, showing columns for 'Status', 'Virtual Host', 'Internal Host', 'Check Result', 'Protocol', 'Back-end Type', and 'Actions'. The table lists entries such as 'abapmessageserver.hana.cloud:sapmsFCB', 'abapserver.hana.cloud:sapgw42', 'asd:123', 'ldciv9u.wdf.sap.corp:sapgw51', 'naboo.empire:8080', 'sccmaster:80', and 'tatooine.empire:443'. Below this table, a section titled 'Resources Of abapmessageserver.hana.cloud:sapmsFCB (4)' is shown, listing 'Function Name' (JTEST\_ADD\_INT, RFC\_RAISE, SBC\_STRING, SFTC\_CONNECTION) and 'Naming Policy' (Exact Name, Prefix, Exact Name, Exact Name). Each item has an 'Actions' column with edit, delete, and copy icons.

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 \(HTTP\)](#).
- For information on configuring RFC resources, see: [Configure Access Control \(RFC\)](#).

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:	<input type="text" value="ABAP System"/>
Protocol:	<input type="text" value="RFC"/>
Virtual Host:	<input type="text" value="abapmessageserver.hana.cloud"/>
Virtual Port:	<input type="text" value="sapmsFCB"/>
Internal Host: <span style="color: red;">*</span>	<input type="text" value="xxxxx.wdf.sap.corp"/>
Internal Port: <span style="color: red;">*</span>	<input type="text" value="sapmsV9U"/>
SAProuter:	<input type="text"/>
Principal Type:	<input type="text" value="None"/>
SNC Partner Name:	<input type="text"/>
Host In Request Header:	<input type="text" value="Use Virtual Host"/>
Description:	<input type="text"/>
Check Internal Host:	<input type="checkbox"/>

Save [Cancel](#)

[Back to Tasks](#)

## Use Scenarios for Resources

As of version 2.12, 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](#)

## [Import a Scenario](#)

## [Remove a Scenario](#)

### 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.

### i 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](#)

[Back to Tasks](#)

## Import a Scenario

As an administrator taking care for 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.

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a navigation sidebar with 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. Under ACCESS CONTROL, there's a table titled 'Mapping Virtual To Internal System (7)' with columns for Status, Virtual Host, Internal Host, Check Result, Protocol, Back-end Type, and Actions. One row is selected, showing 'abapmessageserver.hana.cloud:sapmsFCB'. A modal dialog titled 'Import Scenario' is open over this table. It has a 'Scenario File:' input field with a 'Browse' button, and two buttons at the bottom: 'Import' (highlighted in blue) and 'Cancel'. Below the table, another section titled 'Resources Of abapmessageserver.hana.cloud:sapmsFCB (4)' is visible, showing a table with columns for Status, Function Name, Naming Policy, and Actions. This second table lists four resources: JTEST\_ADD\_INT, RFC\_RAISE, SBC\_STRING, and SFTC\_CONNECTION, each with an 'Exact Name' naming policy.

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](#)

Back to [Tasks](#)

## 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 shows the SAP Cloud Connector Administration interface. On the left, a sidebar for 'Connector' includes options like Security Status, Alerting, High Availability, Hardware Metrics Monitor, and Configuration. A subaccount dropdown is set to 'trial'. The main area displays 'Cloud To On-Premise' settings under 'ACCESS CONTROL'. Below this, two tables are shown: 'Mapping Virtual To Internal System (6)' and 'Resources Of asd:123 (3)'. The first table lists mappings between virtual hosts (e.g., abapmessageserver.hana.cloud:sapms...) and internal hosts (e.g., xxxx.wdf.sap.corp:sapmsV9U), with columns for Status, Virtual Host, Internal Host, Check Result, Protocol, Back-end Type, and Actions. The second table lists resources under 'asd:123' 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"/>	abapmessageserver.hana.cloud:sapms...	xxxx.wdf.sap.corp:sapmsV9U	Unchecked	RFC	ABAP System	
<input type="checkbox"/>	abapserver.hana.cloud:sapgw42	xxxx.wdf.sap.corp:sapgw51	Unchecked	RFC	ABAP System	
<input checked="" type="checkbox"/>	asd:123	asd:123	Unchecked	HTTP	ABAP System	
<input type="checkbox"/>	naboo.empire:8080	xxxx.dhcp.wdf.sap.corp:5555	Unchecked	HTTP	ABAP System	
<input type="checkbox"/>	sccmaster:80	xxxx.wdf.sap.corp:443	Unchecked	HTTP	SAP Business Connector	
<input type="checkbox"/>	tatooine.empire:443	xxxx.dhcp.wdf.sap.corp:4443	Unchecked	HTTP	ABAP System	

Status	URL Path	Access Policy	Actions
<input checked="" type="checkbox"/>	/everybody	Path And All Sub-Paths	
<input checked="" type="checkbox"/>	/public	Path Only (Sub-Paths Are Excluded)	
<input type="checkbox"/>	/qwe	Path And All Sub-Paths	

Back to [Use Scenarios for Resources](#)

Back to [Tasks](#)

## Configuration REST APIs

You can use a set of APIs to perform the basic setup of the Cloud Connector.

### Context

As of version 2.11, 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](#)
- [Managing Subaccounts](#)
- [Configure Access Control](#)

#### i 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](#).

### Available APIs

- [Common Properties](#)
- [High Availability Settings](#)
- [Proxy Settings](#)
- [Authentication and UI Settings](#)
- [Certificate Management for Backend Communication](#)

- [Solution Management Configuration](#)
- [Backup](#)
- [Subaccount](#)
- [System Mappings](#)
- [System Mapping Resources](#)
- [Domain Mappings](#)
- [Subaccount Service Channels](#)
- [Access Control Entities](#)

## Related Information

[Examples](#)

# Common Properties

Read and edit the Cloud Connector's common properties via API.

## Get Common Properties

URI	/api/v1/configuration/connector
Method	GET
Request	
Response	{ha, description}
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

- **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

**i Note**

Available as of version 2.14.0.

URI	/api/v1/connector/version
Method	GET
Request	
Response	

{version}

Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

- **Response Properties:**

version: A string; the Cloud Connector version.

## Set Description (Master Only)

URI	/api/v1/configuration/connector
Method	PUT
Request	{description}
Response	{ha, description}
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.

**i Note**

null is not a JSON string.

## 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](#).

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.

**i Note**

Editing the high availability settings is only allowed on the master instance, and supports only shadow as input.

## Get High Availability Settings

URI	/api/v1/configuration/connector/haRole
Method	<i>GET</i>
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	<i>POST</i>
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 h
```

## Related Information

# Master Instance Configuration

Read and edit the high availability settings for a Cloud Connector master instance via API.

## **i 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, allowedShadowHost}
Errors	
Roles	Administrator, Display, Support

### Response Properties:

- haEnabled: a Boolean value that indicates whether or not a shadow system is allowed to connect
- allowedShadowHost: the 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/maste
```

## Set Configuration

URI	/api/v1/configuration/connector/ha/master/config
Method	PUT
Request	{haEnabled, 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.
- 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}'
```

**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/mast
```

**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"}' ht
```

## 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/m
```

## 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).

**i 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	/api/v1/configuration/connector/ha/shadow/config
Method	GET
Request	
Response	{masterHost, masterPort, ownHost, checkIntervalInSeconds, takeoverDelayInSeconds, connectTimeout}
Errors	
Roles	Administrator, Display, Support

### 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)
- **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)

**i 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/shad
```

## Set Configuration

URI	/api/v1/configuration/connector/ha/shadow/config
Method	PUT
Request	{masterPort, masterHost, ownHost, checkIntervalInSeconds, takeoverDelayInSeconds, connectTimeout}
Response	

	{masterPort, masterHost, ownHost, checkIntervalInSeconds, takeoverDelayInSeconds, connectTimeout}
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)
- **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/shad
-H 'Content-Type: application/json' -d '{"masterHost":"localhost", "masterPort":8443, "ownHost"
"checkIntervalInSeconds":30, "takeoverDelayInSeconds":10, "connectTimeoutInMillis":1000,
"requestTimeoutInMillis":12000}'
```

## Get State

**i 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.

### i 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/shad
```

## Change State

URI	/api/v1/configuration/connector/ha/shadow/state
Method	POST
Request	
Response	{op, user, password}
Errors	INVALID_REQUEST, ILLEGAL_STATE
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.

### i 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/shad -H 'Content-Type: application/json' -d '{"op": "CONNECT", "user": "<user on master>", "pas
```

```
curl -i -k -u <user>:<password> -X POST https://<host>:<port>/api/v1/configuration/connector/ha/sha
-H 'Content-Type: application/json' -d '{"op":"DISCONNECT"}'
```

## Reset

### **i 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	<i>DELETE</i>
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/s
```

## Proxy Settings

Read and edit the Cloud Connector's proxy settings via API.

## Get Proxy Settings

URI	/api/v1/configuration/connector/proxy
Method	<i>GET</i>
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
```

This request removes the proxy configuration.

**↳ Sample Code**

```
curl -ik -u Administrator:<password> https://localhost:8443/api/v1/configuration/connector/proxy
```

## Remove Proxy Settings (Master Only)

URI	/api/v1/configuration/connector/proxy
Method	DELETE
Request	
Response	204 on success
Errors	FORBIDDEN_REQUEST
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
```

## 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	<i>PUT</i>
Request	{password, user}
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	/api/v1/configuration/connector/authentication/basic
Method	<i>PUT</i>
Request	{oldPassword, newPassword}
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
```

## 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	/api/v1/configuration/connector/authentication/ldap
Method	PUT
Request	{enable, configuration}
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).

### **i 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
```

## Get Description for UI Certificate

This API returns a textual description for the system certificate.

**i Note**

Available as of version 2.13.0.

URI	/api/v1/configuration/connector/ui/uiCertificate
Method	GET
Response	{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}
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).

**i 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/c
```

**Create a Self-Signed UI Certificate****i Note**

Available as of version 2.13.0.

URI	/api/v1/configuration/connector/ui/uiCertificate
Method	POST
Request	{type, keySize, subjectDN, subjectAltNames}
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.

**i 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"}' https://<host>:<port>/api/v1/configuration/connector/ui/uiCertificate
```

**Create a Certificate Signing Request for a UI Certificate****i 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.

**i 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.

value (a string). This property is optional.

## Example

```
curl -k -u Administrator:<password> -X POST -H "Content-Type: application/json" --data '{"type":"uiCertificate"}' https://<host>:<port>/api/v1/configuration/connector/ui/uiCertificate -o csr.pem
```

## Upload a Signed Certificate Chain as UI Certificate

### i 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>:<po
```

## 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
keytool -gencert -rfc -infile csr.pem -outfile signedcsr.pem -alias mykey -keystore ca.ks -keypass 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

### i Note

Available as of version 2.13.0.

URI	/api/v1/configuration/connector/ui/uiCertificates
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

#### i 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/uiCertificates
```

## 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 -keystore test20 -keypass test20
```

## Get List of Available Cipher Suites for UI

#### i Note

Available as of version 2.15.0.

Returns a list of available cipher suites (as per JVM).

URI	/api/v1/configuration/availableCipherSuites
-----	---

Method	<i>GET</i>
Request	
Response	[name, ...]
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

**Response Properties:**

List of:

- name: Name of a cipher suite (a string).

**i Note**

name is case-sensitive.

**Example**

```
curl -i -k -H "Accept: application/json" -u <user>:<password> -X GET https://<host>:<port>/api/v1/c
```

**Get List of Enabled Cipher Suites for UI****i Note**

Available as of version 2.15.0.

Returns a list of enabled cipher suites.

URI	/api/v1/configuration/connector/ui/cipherSuites
Method	<i>GET</i>
Request	
Response	[name, ...]
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

**Response Properties:**

List of:

- name: Name of a cipher suite (a string).

**i Note**

name is case-sensitive.

## Example

```
curl -i -k -H "Accept: application/json" -u <user>:<password> -X GET https://<host>:<port>/api/v1/c
```

## Set List of Enabled Cipher Suites for UI

### **i Note**

Available as of version 2.15.0.

URI	/api/v1/configuration/connector/ui/cipherSuites
Method	<i>POST</i>
Request	[name, ...]
Response	204 on success
Errors	
Roles	Administrator

### Request Properties:

List of:

- name: Name of a cipher suite (a string).

### **i Note**

name is case-sensitive.

## Example

```
curl -i -k -H "Content-Type: application/json" -u <user>:<password> --data "[ 'TLS_AES_128_GCM_SHA25
```

## Enable default Cipher Suites for UI

### **i 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).

**i Note**

name is case-sensitive.

**Example**

```
curl -i -k -u <user>:<password> -X DELETE https://<host>:<port>/api/v1/configuration/connector/ui/c
```

## Certificate Management for Backend Communication

Manage a CA certificate for principal propagation or a system certificate via API.

**i 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.

**i 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](#)
- [System Certificate: APIs](#)
- [Truststore CA Certificates](#)

## CA Certificate for Principal Propagation: APIs

Manage a CA certificate for principal propagation via API.

**i Note**

The APIs below are available as of Cloud Connector version 2.13.0.

## Get Description for a CA Certificate for Principal Propagation

URI	/api/v1/configuration/connector/onPremise/ppCaCertificate
Method	GET
Header	Accept: application/json
Response	{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}
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](#) for details).

#### Errors:

- NOT\_FOUND (404): there is no CA certificate for principal propagation.

#### i 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/
```

## Get Binary Content of a CA Certificate for Principal Propagation

URI	/api/v1/configuration/connector/onPremise/ppCaCertificate
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 pppca.crt, for instance, and then running

```
keytool -printcert -file pppca.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://<ho
```

## Create a Self-Signed CA Certificate for Principal Propagation (Master Only)

URI	/api/v1/configuration/connector/onPremise/ppCaCertificate
Method	<i>POST</i>
Request	{type, keySize, subjectDN, subjectAltNames}
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.

#### i 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](#) 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> -
```

## Create a Certificate Signing Request for a CA Certificate for Principal Propagation (Master Only)

URI	/api/v1/configuration/connector/onPremise/ppCaCertificate
Method	<i>POST</i>
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.

**i 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](#) for details). This property is optional.

**Example**

```
curl -k -u <user>:<password> -X POST -H "Content-Type: application/json" --data '{"type":"csr", "su 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	PATCH
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>:<po
```

**Example: Sign The Certificate Signing Request**

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

```
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

### i 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/ppCa
```

Create an own self-signed pkcs#12 certificate for tests with:

```
keytool -genkeypair -alias key -keyalg RSA -keysize 2048 -validity 365 -keystore test20 -keypass testit -storepass testit
```

## Delete a CA Certificate for Principal Propagation (Master Only)

URI	/api/v1/configuration/connector/onPremise/ppCaCertificate
Method	DELETE
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/
```

## System Certificate: APIs

Manage a system certificate via API.

**i Note**

The APIs below are available as of Cloud Connector version 2.13.0.

### Get Description for a System Certificate

URI	/api/v1/configuration/connector/onPremise/systemCertificate
Method	GET
Header	Accept: application/json
Response	{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}
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](#) for details).

**Errors:**

- NOT\_FOUND (404): there is no system certificate.

**i 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/
```

### 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://
```

### 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, subjectAltNames}
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.

**i 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](#) 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> -
```

## 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, 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.

**i 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](#) for details). This property is optional.

## Example

```
curl -k -u <user>:<password> -X POST -H "Content-Type: application/json" --data '{"type":"csr", "su https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate -o csr.pem'}
```

## Upload a Signed Certificate Chain as System Certificate (Master Only)

URI	/api/v1/configuration/connector/onPremise/systemCertificate
Method	PATCH
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>
```

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
keytool -gencert -rfc -infile csr.pem -outfile signedcsr.pem -alias mykey -keystore ca.ks -keypass 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

**i 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/syst
```

Create an own self-signed pkcs#12 certificate for tests with:

```
keytool -genkeypair -alias key -keyalg RSA -keysize 2048 -validity 365 -keystore test20 -keypass test20
```

## Delete a System Certificate (Master Only)

URI	/api/v1/configuration/connector/onPremise/systemCertificate
Method	DELETE
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/
```

## Truststore CA Certificates

Manage CA certificates in the truststore via API.

## Get Truststore Configuration

**i 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).

**i 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/truststore
```

**Change Truststore Configuration (Master Only)****i Note**

This API is available as of Cloud Connector version 2.15.0.

URI	/api/v1/configuration/connector/onPremise/truststore
Method	PATCH
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}'
```

## Add a CA Certificate to 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</code>
Method	<code>POST</code>
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/configur
```

## 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/&lt;alias&gt;</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/
```

**Delete All CA Certificates From Truststore (Master Only)****i Note**

This API is available as of Cloud Connector version 2.14.1.

URI	/api/v1/configuration/connector/onPremise/truststore/certificates
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/
```

**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
```

or, if configuration has to be changed:

```
curl -ik -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/solutionManagement
```

**Turn Off Solution Management Reporting**

URI	/api/v1/configuration/connector/solutionManagement
Method	DELETE

Request	
Response	
Errors	
Roles	Administrator, Support

## Example

```
curl -ik -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/solutionManagement/registrationFile
```

## Download Current LMDB XML Report

Generates a zip file containing the registration file for the solution management LMDB (Landscape Management Database).

**i 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

## 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	
Roles	Administrator

### Request Properties:

- password: the password used to encrypt sensitive data.

**i 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
```

## 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
Roles	Administrator

### Request Properties:

- **backup**: a backup file (produced through POST request).
- **password**: the password chosen when creating the backup.

### Errors:

- INVALID\_REQUEST (400): invalid or missing file, or incorrect or missing password.

**i 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
```

## Subaccount

Manage the Cloud Connector's subaccount settings via API.

## Operations

<b>Subaccount</b>	<a href="#">Get Subaccounts</a> <a href="#">Create Subaccount (Master Only)</a> <a href="#">Delete Subaccount (Master Only)</a> <a href="#">Edit Subaccount (Master Only)</a> <a href="#">Connect/Disconnect Subaccount (Master Only)</a> <a href="#">Refresh Subaccount Certificate (Master Only)</a> <a href="#">Get Subaccount Configuration</a>
<b>Recovery Subaccount</b>	<a href="#">Create Recovery Subaccount (Master Only)</a> <a href="#">Delete Recovery Subaccount (Master Only)</a> <a href="#">Refresh Certificate of Recovery Subaccount (Master Only)</a> <a href="#">Activate/Deactivate Recovery Subaccount (Master Only)</a> <a href="#">Takeover (Master Only)</a>
<b>Trust</b>	<a href="#">Synchronize Trust List (Master Only)</a> <a href="#">Get IDP Trust List</a> <a href="#">Get Application Trust List</a> <a href="#">Enable Or Disable Trust (Master Only)</a> <a href="#">Get IDP Trust Properties</a> <a href="#">Get Application Trust Properties</a>

## Get Subaccounts

URI	/api/v1/configuration/subaccounts
Method	GET
Request	
Response	[{regionHost, subaccount, locationID}]
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](#)

## Create Subaccount (Master Only)

Creates and connects a subaccount.

URI	/api/v1/configuration/subaccounts
Method	<i>POST</i>
Request	{regionHost, subaccount, cloudUser, cloudPassword, locationID, displayName, description}
Response	201, created subaccount entity:  {regionHost, subaccount, locationID, displayName, description, tunnel}
Errors	INVALID_REQUEST, INVALID_CONFIGURATION
Roles	Administrator, Subaccount Administrator

### 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 as a regular subaccount or recovery subaccount.

Back to [Operations](#)

## Delete Subaccount (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>
Method	<i>DELETE</i>
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](#)

## Edit Subaccount (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>
Method	<i>PUT</i>
Request	{locationID, displayName, description}
Response	{regionHost, subaccount, locationID, displayName, description, tunnel, recoveryAccountState}
Errors	NOT_FOUND
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.
- **recoveryAccountState:** object detailing the current state of the recovery subaccount. This property is supplied only if a recovery subaccount is configured.

## Errors

- NOT\_FOUND (404): subaccount does not exist (in the specified region).

Back to [Operations](#)

## Connect/Disconnect Subaccount (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/state
Method	PUT
Request	{connected}
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](#)

## Refresh Subaccount Certificate (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/validity
Method	POST
Request	{user, password}
Response	{regionHost, subaccount, locationID, displayName, description, tunnel, recoveryAccountState}
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.
- **recoveryAccountState**: object detailing the current state of the recovery subaccount. This property is supplied only if a recovery subaccount is configured.

Back to [Operations](#)

## Get Subaccount Configuration

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>
Method	<i>GET</i>
Request	
Response	{regionHost, subaccount, locationID, displayName, description, tunnel:{state, connections, appli}
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**: object outlining the current state of the tunnel.

Back to [Operations](#)

## Create Recovery Subaccount (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery
Method	<i>POST</i>

Request	{regionHost, cloudUser, cloudPassword}
Response	201 on success
Errors	INVALID_REQUEST, ILLEGAL_STATE
Roles	Administrator, Subaccount Administrator

**Request Properties:**

- **regionHost**: region host of the recovery subaccount.
- **cloudUser**: user for the specified region host and recovery subaccount.
- **cloudPassword**: password for the cloud user.

**Errors:**

- **INVALID\_REQUEST** (400): invalid or missing request parameters.
- **ILLEGAL\_STATE** (409): a recovery subaccount already exists.

**i Note**

A recovery subaccount cannot be changed. Delete it and create a new one instead.

Back to [Operations](#)

## Delete Recovery Subaccount (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery
Method	<i>DELETE</i>
Request	
Response	
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

**Errors:**

- **NOT\_FOUND** (404): there is no recovery subaccount.

Back to [Operations](#)

## Refresh Certificate of Recovery Subaccount (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/validity
Method	<i>POST</i>
Request	{user, password}

Response	{regionHost, subaccount, locationID, displayName, description, tunnel, recoveryAccountState}
Errors	INVALID_REQUEST, NOT_FOUND
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.
- **recoveryAccountState**: object detailing the current state of the recovery subaccount. This property is supplied only if a recovery subaccount is configured.

**Errors**

- **INVALID\_REQUEST** (400): user or password are missing.
- **NOT\_FOUND** (404): there is no recovery subaccount.

Back to [Operations](#)

## Activate/Deactivate Recovery Subaccount (Master Only)

**i Note**

Available as of Cloud Connector 2.13.1.

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/state
Method	PUT
Request	{active}
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

**Request Properties:**

- **active**: Boolean value indicating whether the recovery subaccount should be active (true) or inactive (false).

**Errors:**

- NOT\_FOUND (404): there is no recovery subaccount.

Back to [Operations](#)

## Takeover (Master Only)

**⚠ Caution**

When performing this operation, the recovery subaccount *permanently takes over* from the original subaccount, and the *original subaccount is deleted*. The *recovery subaccount must be active* for takeover to succeed, see [Activate/Deactivate Recovery Subaccount \(Master Only\)](#).

**i Note**

Available as of Cloud Connector 2.13.1.

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/takeover
Method	POST
Request	
Response	204 on success
Errors	NOT_FOUND, ILLEGAL_STATE
Roles	Administrator, Subaccount Administrator

**Errors:**

- NOT\_FOUND (404): there is no recovery subaccount.
- ILLEGAL\_STATE (409): recovery subaccount is not active.

Back to [Operations](#)

## Synchronize Trust List (Master Only)

**i Note**

Available as of version 2.14.0.

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/trust
Method	POST
Request	

Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Back to [Operations](#)

## Get IDP Trust List

### i Note

Available as of version 2.14.0.

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/trust/idps
Method	<i>GET</i>
Request	
Response	[{id, name, description, certificate, enabled}]
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](#)

## Get Application Trust List

### i Note

Available as of version 2.14.0.

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/trust/apps
Method	<i>GET</i>
Request	
Response	

	[{id, name, applicationType, enabled}]
--	--

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](#)

## Enable Or Disable Trust (Master Only)

Replace <type> with the type (either *apps* or *idps*) that belongs to the specified <id>.

#### i Note

Available as of version 2.14.0.

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/trust/<type>/<id>
Method	PUT
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](#)

## Get IDP Trust Properties

#### i Note

Available as of version 2.14.0.

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/trust/idps/<id>
Method	GET

Request	
Response	{id, name, description, certificate, enabled}
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](#)

## Get Application Trust Properties

**i Note**

Available as of version 2.14.0.

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/trust/apps/<id>
Method	GET
Request	
Response	{id, name, applicationType, enabled}
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](#)

# System Mappings

Manage the Cloud Connector's system mappings via API.

## Get All System Mappings

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings
Method	<i>GET</i>
Request	
Response	[{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader, totalResourcesCount, enabledResourcesCount, description, sncPartnerName, sapRouter, allowedClients}]
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.

- `blacklistedClientUsers`: 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	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/<virtualHost>:<\
Method	<i>GET</i>
Request	
Response	{virtualHost, virtualPort, localhost, localPort, protocol, backendType, authenticationMode, hostInHeader, totalResourcesCount, enabledResourcesCount, description, sncPartnerName, sapRouter, allowedClients, blacklistedClientUsers}
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.
- `blacklistedClientUsers`: 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	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings
-----	--

Method	<i>POST</i>
Request	{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader}
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.
- **blacklistedClientUsers**: 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.

## Delete System Mapping (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

## Delete All System Mappings (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

## Replace System Mapping

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/virtualHost:virt
Method	<i>PUT</i>
Request	{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader}
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`, `appServerJava`, `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.
- **blacklistedClientUsers**: 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.

# System Mapping Resources

Manage the Cloud Connector's system mapping resources via API.

## Operations

<b>System Mapping Resources</b>	<a href="#">Get all System Mapping Resources</a> <a href="#">Get System Mapping Resource</a> <a href="#">Create System Mapping Resource</a> <a href="#">Edit System Mapping Resource</a> <a href="#">Delete System Mapping Resource</a> <a href="#">Delete all System Mapping Resources</a>
<b>Allowed Clients</b>	<a href="#">Get Allowed Clients for RFC System Mapping</a> <a href="#">Set Allowed Clients for RFC System Mapping</a> <a href="#">Add Allowed Clients for RFC System Mapping</a> <a href="#">Delete an Allowed Client for RFC System Mapping</a> <a href="#">Delete all Allowed Clients for RFC System Mapping</a>
<b>Blocked Users</b>	<a href="#">Get Blocked Users for RFC System Mapping</a> <a href="#">Set Blocked Users for RFC System Mapping</a> <a href="#">Add Blocked Users for RFC System Mapping</a> <a href="#">Remove one Blocked User for RFC System Mapping</a> <a href="#">Remove all Blocked Users for RFC System Mapping</a>

## Get all System Mapping Resources

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/virtualHost:virtualPort/resources
Method	<i>GET</i>
Request	
Response	[{id, enabled, exactMatchOnly, websocketUpgradeAllowed, description}]
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](#)

## Get System Mapping Resource

URI	<code>/api/v1/configuration/subaccounts/&lt;regionHost&gt;/&lt;subaccount&gt;/systemMappings/virtualHost:virtualPort/resources/&lt;encodedResourceId&gt;</code>
Method	<code>GET</code>
Request	
Response	<code>{id, enabled, exactMatchOnly, websocketUpgradeAllowed, description}</code>
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](#)

## Create System Mapping Resource

URI	<code>/api/v1/configuration/subaccounts/&lt;regionHost&gt;/&lt;subaccount&gt;/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 '-'.

Back to [Top](#)

**Replace System Mapping Resource**

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/virtualHost:virtualPort/resources/<encodedResourceId>
Method	<i>PUT</i>
Request	{enabled, exactMatchOnly, websocketUpgradeAllowed, description}
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](#)

## 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](#)

## 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](#)

## Get Allowed Clients for RFC System Mapping

**i Note**

Available as of version 2.1.4.

Returns the list of allowed ABAP clients for the system mapping definition.

URI	/api/v1/configuration/subaccounts/<region>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>/allowedClients
Method	<i>GET</i>
Request	
Response	[client, ...]
Errors	

**Response Properties:**

- Array of allowed clients:
  - client: ABAP client

**i Note**

An empty list means that every client is allowed.

Back to [Top](#)

## Set Allowed Clients for RFC System Mapping

**i Note**

Available as of version 2.1.4.

URI	/api/v1/configuration/subaccounts/<region>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>/allowedClients
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 request.

Back to [Top](#)

## Add Allowed Clients for RFC System Mapping

**i 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](#)

## Delete an Allowed Client for RFC System Mapping

**i Note**

Available as of version 2.1.4.

URI	/api/v1/configuration/subaccounts/<region>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>/allowedClients/<client>
Method	<i>DELETE</i>
Request	
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Back to [Top](#)

## Delete all Allowed Clients for RFC System Mapping

**i Note**

Available as of version 2.1.4.

URI	/api/v1/configuration/subaccounts/<region>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>/allowedClients
Method	<i>DELETE</i>
Request	
Response	
Errors	

Roles	Administrator, Subaccount Administrator, Display, Support
-------	---

**i Note**

An empty list means that every client is allowed.

Back to [Top](#)

## Get Blocked Users for RFC System Mapping

**i Note**

Available as of version 2.1.4.

Returns the list of blocked ABAP users for the system mapping definition.

URI	/api/v1/configuration/subaccounts/<region>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>/blockedClientUsers
Method	<i>GET</i>
Request	
Response	[{client, user}, ...]
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

### Response Properties:

- Array of blocked users:
  - client: ABAP client
  - user: User name

Back to [Top](#)

## Set Blocked Users for RFC System Mapping

**i Note**

Available as of version 2.1.4.

Sets or replaces the list of blocked ABAP users for the system mapping definition.

URI	/api/v1/configuration/subaccounts/<region>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>/blockedClientUsers
Method	<i>POST</i>
Request	[{client, user}, ...]

Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

**Request Properties:**

- Array of blocked users:
  - client: ABAP client
  - user: User name

Back to [Top](#)

## Add Blocked Users for RFC System Mapping

**i Note**

Available as of version 2.1.4.

Adds the provided list of blocked ABAP users to the current list.

URI	/api/v1/configuration/subaccounts/<region>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>/blockedClientUsers
Method	PATCH
Request	[{client, user}, ...]
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

**Request Properties:**

- Array of blocked users:
  - client: ABAP client
  - user: User name

Back to [Top](#)

## Remove one Blocked User for RFC System Mapping

**i Note**

Available as of version 2.1.4.

Removes one user from the list of blocked ABAP users for the system mapping definition.

URI	/api/v1/configuration/subaccounts/<region>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>/blockedClientUsers/<client>:<user>
Method	<i>DELETE</i>
Request	
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Back to [Top](#)

## Remove all Blocked Users for RFC System Mapping

### i Note

Available as of version 2.1.4.

Removes the list of blocked ABAP users for the system mapping definition.

URI	/api/v1/configuration/subaccounts/<region>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>/blockedClientUsers
Method	<i>DELETE</i>
Request	
Response	
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Back to [Top](#)

## 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	<i>GET</i>
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	<i>POST</i>
Request	{virtualDomain, internalDomain}
Response	201
Errors	
Roles	Administrator, Subaccount Administrator

## Replace Domain Mapping (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/domainMappings/<internalDomain>
Method	<i>PUT</i>
Request	{virtualDomain, internalDomain}
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.

**i 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	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/domainMappings/<internalDomain>
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	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/domainMappings
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

## Subaccount Service Channels

Manage Cloud Connector service channels via API.

<b>Service Channel for HANA Database</b>	<a href="#">Get all HANA Service Channels</a> <a href="#">Get Available HANA Channels</a> <a href="#">Get HANA Service Channel</a> <a href="#">Create HANA Service Channel (Master Only.)</a> <a href="#">Replace HANA Service Channel (Master Only.)</a>
<b>Service Channel for Virtual Machine</b>	<a href="#">Get all Service Channels for Virtual Machines</a> <a href="#">Get Available Channels for Virtual Machines</a> <a href="#">Get Service Channel for a Virtual Machine</a> <a href="#">Create a Service Channel for a Virtual Machine (Master Only.)</a> <a href="#">Replace a Service Channel for a Virtual Machine (Master Only.)</a>
<b>Service Channel for ABAP Cloud</b>	<a href="#">Get all ABAP Cloud Service Channels</a> <a href="#">Get ABAP Cloud Service Channel</a> <a href="#">Create ABAP Cloud Service Channel (Master Only.)</a> <a href="#">Replace ABAP Cloud Service Channel (Master Only.)</a>
<b>Service Channel for Kubernetes</b>	<a href="#">Get All Kubernetes Cluster Service Channels</a>

**i Note**

Available as of version 2.15.0.

[Get Kubernetes Service Channel](#)

[Create Kubernetes Service Channel \(Master Only\)](#)

[Replace Kubernetes Service Channel \(Master Only\)](#)

*General*

[Enable/Disable Service Channel \(Master Only\)](#)

[Delete Service Channel \(Master Only\)](#)

[Delete all Service Channels \(Master Only\)](#)

*Deprecated*

[Get all Service Channels](#)

[Get Service Channel](#)

[Create Service Channel](#)

[Replace Service Channel](#)

## Get all HANA Service Channels

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/HANA
Method	<i>GET</i>
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).

## Get Available HANA Channels

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/availableChannels/HANA
Method	GET
Request	
Response	[{hanaInstanceName, version, type}]
Errors	RUNTIME_FAILURE
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.

## Get HANA Service Channel

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/HANA/<id>
Method	GET
Request	
Response	{id, hanaInstanceName, instanceNumber, 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.
- **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](#)

## Create HANA Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/HANA
Method	<i>POST</i>
Request	{hanaInstanceName, instanceNumber, connections}
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](#)

## Replace HANA Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/HANA/<id>
Method	<i>PUT</i>
Request	{hanaInstanceName, instanceNumber, connections}

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](#)

## Get all Service Channels for Virtual Machines

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/VirtualMachine
Method	GET
Request	
Response	[{id, name, endpointId, type, port, enabled, connections, state}]
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](#)

## Get Available Channels for Virtual Machines

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/availableChannels/VirtualMachine
Method	<i>GET</i>
Request	
Response	[{endpointId, name}]
Errors	RUNTIME_FAILURE
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](#)

## Get Service Channel for a Virtual Machine

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/VirtualMachine/<id>
Method	<i>GET</i>
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](#)

## 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](#)

## 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](#)

## Get all ABAP Cloud Service Channels

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/ABAPCloud
Method	GET
Request	
Response	[{id, abapCloudTenantHost, instanceNumber, type, port, enabled, connections, state}]
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](#)

## Get ABAP Cloud Service Channel

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/ABAPCloud/<id>
Method	<i>GET</i>
Request	
Response	{ <b>id</b> , <b>abapCloudTenantHost</b> , <b>instanceNumber</b> , <b>type</b> , <b>port</b> , <b>enabled</b> , <b>connections</b> , <b>state</b> }
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](#)

## Create ABAP Cloud Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/ABAPcloud
Method	<i>POST</i>
Request	{abapCloudTenantHost, instanceNumber, connections}
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](#)

## Replace ABAP Cloud Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/ABAPcloud/<id>
Method	<i>PUT</i>
Request	{abapCloudTenantHost, instanceNumber, connections}
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](#)

## Get All Kubernetes Cluster Service Channels

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/K8S
Method	GET
Request	
Response	[{id, type, port, k8sCluster, k8sService, enabled, connections, state, comment}]
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](#)

## Get Kubernetes Service Channel

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/K8S/<id>
Method	GET
Request	
Response	[{id, type, port, k8sCluster, k8sService, enabled, connections, state, comment}]

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](#)

## Create Kubernetes Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/K8S
Method	<i>POST</i>
Request	{k8sCluster, k8sService, port, connections, comment}
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.

## 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](#)

## 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	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/<type>/<id>/state
Method	PUT
Request	{enabled}
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](#)

## 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	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/<type>/<id>
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](#)

## 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	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels[/<type>]
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Back to [Top](#)

## 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	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels
Method	<i>GET</i>
Request	
Response	[{typeDesc, details, port, enabled, connected, connectionsCount, availableConnectionsCount, conn}
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](#)

## 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	<i>GET</i>
Request	
Response	

	[{typeDesc, details, port, enabled, connected, connectionsCount, availableConnectionsCount, conn}
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](#)

## 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	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels
Method	<i>POST</i>
Request	{typeKey, details, serviceNumber, connectionCount}
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](#)

## 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/&lt;regionHost&gt;/&lt;subaccount&gt;/channels/&lt;id&gt;</code>
Method	<code>PUT</code>
Request	<code>{typeKey, details, serviceNumber, connectionCount}</code>
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Back to [Top](#)

## Access Control Entities

Manage RFC-specific access control entities for the Cloud Connector via API.

[Get List Of Allowed Clients](#)

[Get List Of Blocked Client/User Pair](#)

[Create Allowed Clients \(Master Only\)](#)

[Extend Allowed Clients \(Master Only\)](#)

[Create a Blocked Client/User Pair \(Master Only\)](#)

[Extend a Blocked Client/User Pair \(Master Only\)](#)

[Delete All Allowed Clients \(Master Only\)](#)

[Delete All Blocked Client/User Pairs \(Master Only\)](#)

[Remove an Item from the List of Allowed Clients \(Master Only\)](#)

[Remove Items on the List of Blocked Client/User Pairs for a Given User \(Master Only\)](#)

[Remove Items on the List of Blocked Client/User Pairs for a Given Client \(Master Only\)](#)

[Remove an Item on the List of Blocked Client/User Pairs \(Master Only\)](#)

## Get List Of Allowed Clients

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/allowedClients
Method	<i>GET</i>
Request	
Response	[{client}]
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

### Response:

List of objects, each representing an allowed client:

- **client**: client name

Back to [Top](#)

## Get List Of Blocked Client/User Pair

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/blockedClientUsers
Method	<i>GET</i>
Request	
Response	[{client, user}]
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](#)

## Create Allowed Clients (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/allowedClients
Method	<i>POST</i>

Request	[client]
Response	201 on success
Errors	
Roles	Administrator, Subaccount Administrator

**Request:**

List of strings, each representing an allowed client.

Back to [Top](#)

## Extend Allowed Clients (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/allowedClients
Method	<i>PATCH</i>
Request	[client]
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

**Request:**

List of strings, each representing an allowed client.

Back to [Top](#)

## Create a Blocked Client/User Pair (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/blockedClientUsers
Method	<i>POST</i>
Request	[{client, user}]
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](#)

## Extend a Blocked Client/User Pair (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/blockedClientUsers
Method	<i>PATCH</i>
Request	[{client, user}]
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](#)

## Delete All Allowed Clients (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/allowedClients
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Back to [Top](#)

## Delete All Blocked Client/User Pairs (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/blockedClientUsers
Method	<i>DELETE</i>
Request	

Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Back to [Top](#)

## Remove an Item from the List of Allowed Clients (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/allowedClients/<client>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Back to [Top](#)

## Remove Items on the List of Blocked Client/User Pairs for a Given User (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/blockedClientUsers/user/<user>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Back to [Top](#)

## Remove Items on the List of Blocked Client/User Pairs for a Given Client (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/blockedClientUsers/client/<client>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

## Remove an Item on the List of Blocked Client/User Pairs (Master Only)

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMapping/<virtualHost>:<virtualPort>/blockedClientUsers/client/<client>:<user>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

[Back to Top](#)

## 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](#)

[REST API Invocation](#)

[How To Use the Examples](#)

## 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](#)

## REST API Invocation

```
(a)     Fuel.put(url)
(b)     .header("Connection", "close")
(c)     .authentication().basic(user, password)
(d)     .jsonBody(credentials)
(e)     .responseObject<OnlyPropertiesNamesAreRelevant> { _, _, result ->
            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](#).

[Back to Concept](#)

## 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](#). This is the only help class under sources/.

When using the examples, start with [Initial Configuration](#).

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](#))
- Create or restore a configuration backup ([Backup And Restore Configuration](#))
- Configure and connect high availability instances ([High Availability Settings](#))
- Integrate the Cloud Connector with the solution management infrastructure ([Solution Management Integration](#))

Back to [Concept](#)

## Related Information

[scenario.json](#)

[Source Files](#)

## 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": "Administrator"
  }
}
```

```

        "password": "test"
    },
    "shadow": {
        "url": "https://localhost:8444",
        "user": "Administrator",
        "password": "test"
    }
}

```

## Source Files

[Scenario Configuration](#)

[Initial Configuration](#)

[Subaccount Configuration](#)

[High Availability Settings](#)

[Backup And Restore Configuration](#)

[Solution Management Integration](#)

## 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 {
        HttpsURLConnection.setDefaultHostnameVerifier { hostname: String, session: SSLSession ->
            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())
        }
    } catch (e: Exception) {
        e.printStackTrace()
    }
}

```

```

       HttpsURLConnection.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,
    var locationId: String? = null
)
data class SccCertificate(
    var subjectDN: String? = null,
    var issuer: String? = null,
    var notAfter: String? = null,
    var notBefore: String? = null,
    var subjectAltNames: List<SubjectAltName>? = null
)
data class SubjectAltName(
    var type: String? = null,
    var 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. ")
}

```

## 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,
and upload UI and system certificates.

For the certificates used by Cloud Connector in order to access the UI and for the system certificates
we simply upload the already available PKCS#12 certificates uiCert.p12 and systemCert.p12 encrypted.
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(
    "${config.master!!.url}/api/v1/configuration/connector/ui/uiCertificate",
    Method.PUT,
    uiCertificateFormData
)
    .add(BlobDataPart(ByteArrayInputStream(File("uiCert.p12").readBytes()), name = "uiCert.p12"))
    .authentication().basic(config.master!!.user, config.master!!.password)
    .response { _, _, result ->
        when (result) {
            is Result.Failure -> processRequestError(result.error)
            is Result.Success -> println("PKCS#12 Certificate 'uiCert.p12' uploaded")
        }
    }
    .join()
//Upload a PKCS#12 Certificate as System Certificate
val systemCertificateFormData = listOf("password" to "test1234", "keyPassword" to "test1234")
Fuel.upload(
    "${config.master!!.url}/api/v1/configuration/connector/onPremise/systemCertificates",
    Method.PUT,
    systemCertificateFormData
)
    .add(BlobDataPart(ByteArrayInputStream(File("systemCert.p12").readBytes()), name = "systemCert.p12"))
    .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' uploaded")
        }
    }
    .join()
}
//Data structures used by REST calls in this scenario
data class SccDescriptionResponse(
    var role: String,
    var description: String
)

```

## 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 HT

```

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
    )
    //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 :
    //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!!!.passw
        .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>
                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)
    }
}
}

//2.2 Add an allowed resource to the system mapping. Since this system mapping points to an I
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>,
                //<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>,
        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>
        //<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 = ""
)

```

## 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.
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
    //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: $masterConfig ")
                }
            }
        }
        .join()
    //The configuration of the shadow instance
    var shadowConfig: HASHadowConfiguration? = null
    //Get the current configuration
    Fuel.get("${config.shadow!!.url}/api/v1/configuration/connector/ha/shadow/config")
        .authentication().basic(config.shadow!!.user, config.shadow!!.password)
        .responseObject<HASHadowConfiguration> { _, _, 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<HAShadowConfiguration> { _, _, 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":"${config.master!.
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("Master (${config.master!!!.url}) and shadow (${conf:
            }
        }
        .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?
    )
)

```

## 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.
As a prerequisite you have a stable Cloud Connector configuration and you want to save the backup.

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()
}

```

## 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.
    Fuel.get("${config.master!!}.url}/api/v1/configuration/connector/solutionManagement/registration")
        .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
)

```

## Using Service Channels

Configure Cloud Connector service channels to connect your on-premise network to specific services on SAP BTP.

## Context

Cloud Connector service channels provide access from an external network to certain services on SAP BTP. 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
SAP HANA Database on SAP BTP	The service channel for the <b>SAP HANA Database</b> lets you access SAP HANA databases that run in the Cloud from database clients (for example, clients using ODBC/JDBC drivers). You can use the service channel to connect database, analytical, BI, or replication tools to your SAP HANA database in your SAP BTP subaccount.
RFC Connection to SAP BTP ABAP environment	The service channel for <b>RFC</b> supports calls from on-premise systems to the SAP BTP ABAP environment 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 an SAP HANA Database](#)

[Connect DB Tools to SAP HANA via Service Channels](#)

[Configure a Service Channel for RFC](#)

[Configure a Service Channel for a Kubernetes Cluster](#)

## Related Information

[Service Channels: Port Overview](#)

## Configure a Service Channel for an SAP HANA Database

Using Cloud Connector service channels, you can establish a connection to an SAP HANA database in SAP BTP that is not directly exposed to external access.

## Context

The service channel for SAP HANA Database lets you access SAP HANA databases running in the cloud via ODBC/JDBC. You can use the service channel to connect database, analytical, BI, or replication tools to an SAP HANA database in your SAP BTP subaccount.

### Restrictions

This feature is only available in regions (data centers) that provide the SAP HANA service (that is, SAP data centers (Neo environment), Azure and AWS).

For more information, see [SAP HANA Service for SAP BTP Getting Started Guide](#).

- In AWS regions, you can currently use the service channel only for the SAP HANA service provisioned *before June 4, 2018* (old version).

- If you are using the SAP HANA service provisioned *after June 4, 2018* (new version, available on AWS and Google Cloud) or the *SAP HANA Cloud service*, you can currently access SAP HANA databases only directly, without going through the Cloud Connector.

For more information, see [Connecting to an SAP HANA Service Instance Directly from SAP HANA Clients](#).

To find detailed information for a specific Cloud Foundry region and SAP HANA service version, see [Find the Right Guide](#).

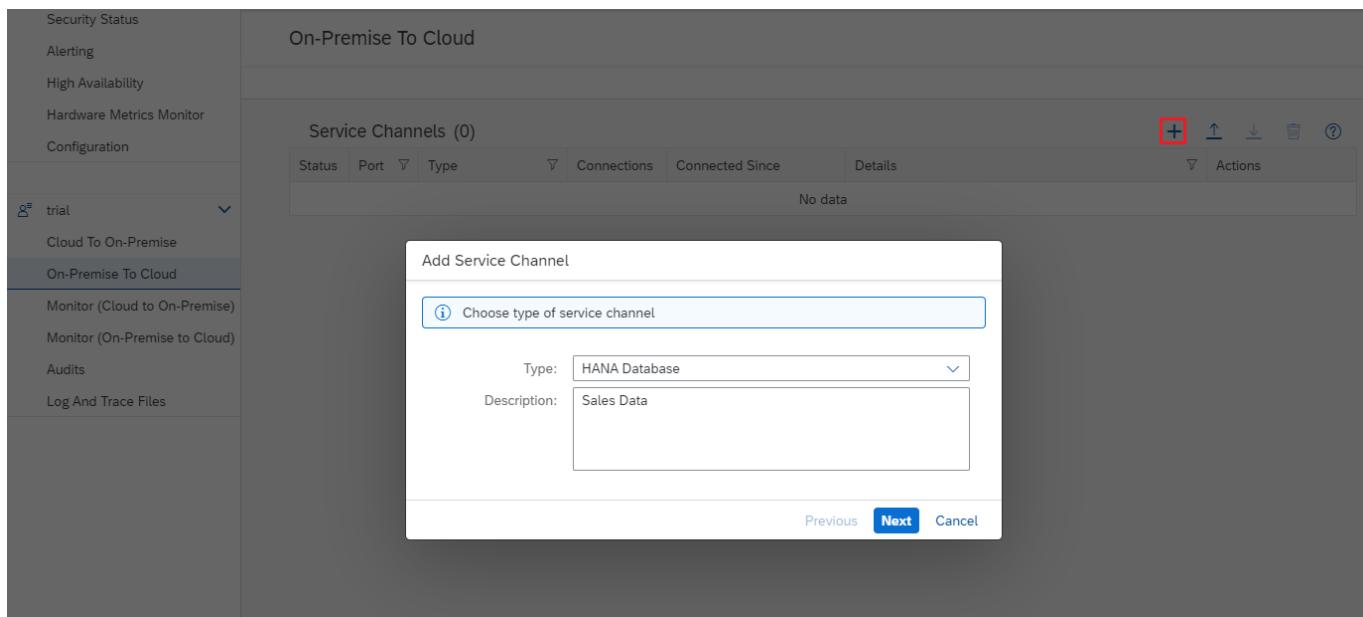
## i Note

The following procedure requires a productive SAP HANA instance that is available in the *same* subaccount. You cannot access an SAP HANA instance that is owned by a different subaccount within the same or another global account (shared SAP HANA database). See also [Sharing Databases with Other Subaccounts](#).

## Procedure

- From your subaccount menu, choose **On-Premise To Cloud**.

- Choose **Add (+)**.



- In the **Add Service Channel** dialog, leave the default value **HANA Database** in the **<Type>** field.

- Optionally, provide a **Description** that explains what the HANA DB service channel is used for.

- Choose **Next**.

- Choose the SAP HANA instance name. If you cannot select it from the drop-down list, enter the instance name manually. In the **Neo** environment, it must match one of the names (IDs) shown in the cockpit under **SAP HANA/SAP ASE > Databases & Schemas**, in the **<DB/Schema ID>** column.

## i Note

The SAP HANA instance name is case-sensitive.

In the **Cloud Foundry** environment, the format of the SAP HANA instance name includes the Cloud Foundry space name, the database name and the database ID.

**Example:**

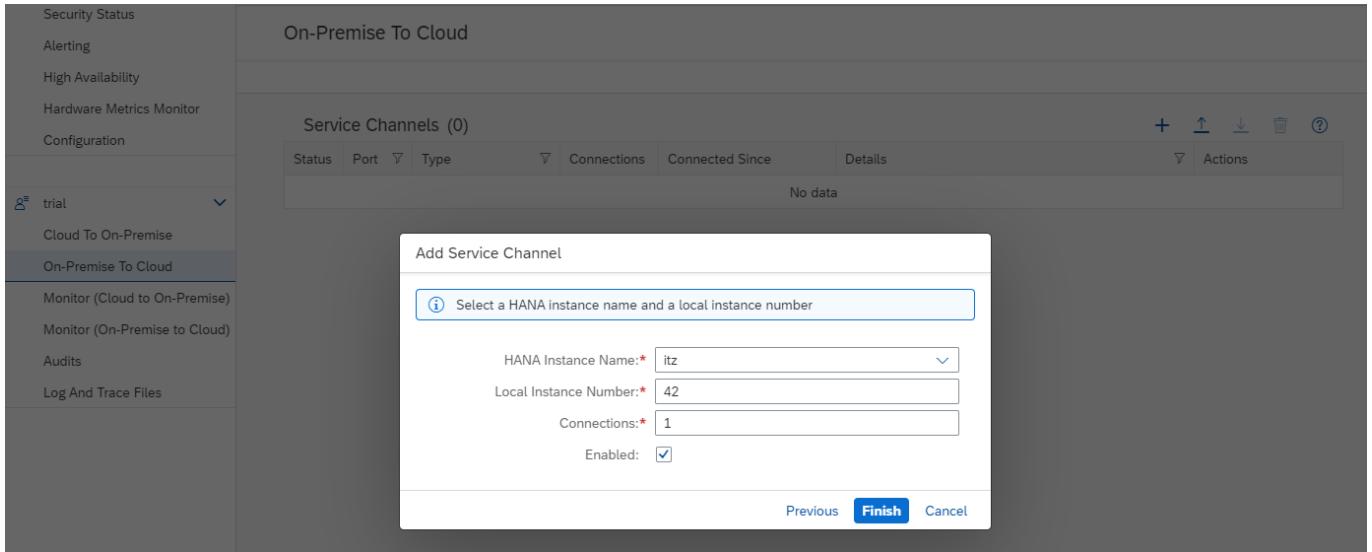
**test:testInstance:3fcc976d-457a-474e-975b-e572600f474e:de19c262-a1fc-4096-bfce-1c41388e4b49**

where

- **test**: Cloud Foundry space name
- **testInstance**: tenant database instance name
- **3fcc976d-457a-474e-975b-e572600f474e:de19c262-a1fc-4096-bfce-1c41388e4b49**: database ID

7. Specify the local instance number. This is a double-digit number which computes the local port used to access the SAP HANA instance in the cloud. The local port is derived from the local instance number as  $3 < \text{instance number} > 15$ . For example, if the instance number is 22, then the local port is 32215. The local instance number must not be 00. This instance number might cause problems with some SAP HANA clients.

8. Specify the number of physical connections to SAP BTP. The default value is 1. One physical connection can open various virtual connections through multiplexing.



9. Leave **Enabled** selected to establish the channel immediately after clicking **Finish**, or unselect it if you don't want to establish the channel immediately.

10. Choose **Finish**.

## Next Steps

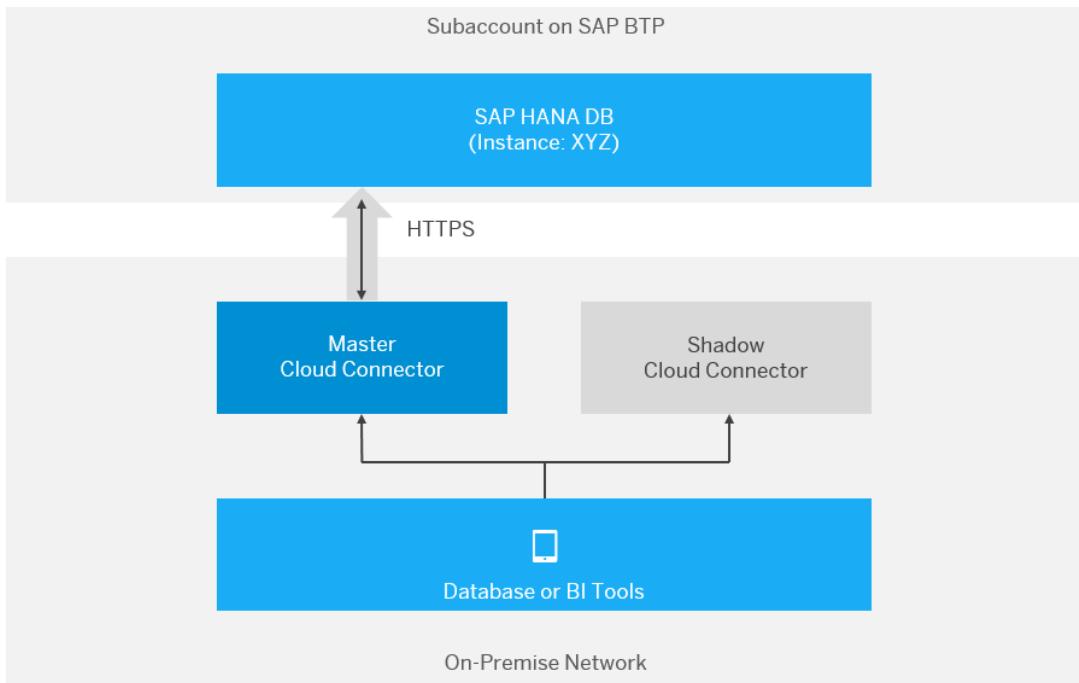
Once you have established an SAP HANA Database service channel, you can connect on-premise database or BI tools to the selected SAP HANA database in the cloud. This can be done by using `<cloud_connector_host>:<local_HANA_port>` in the JDBC/ODBC connect strings.

See [Connect DB Tools to SAP HANA via Service Channels](#).

## 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](#).

### i 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](#).
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,<c>
```

## Related Information

[Guides for earlier releases of SAP HANA](#)

# Configure a Service Channel for RFC

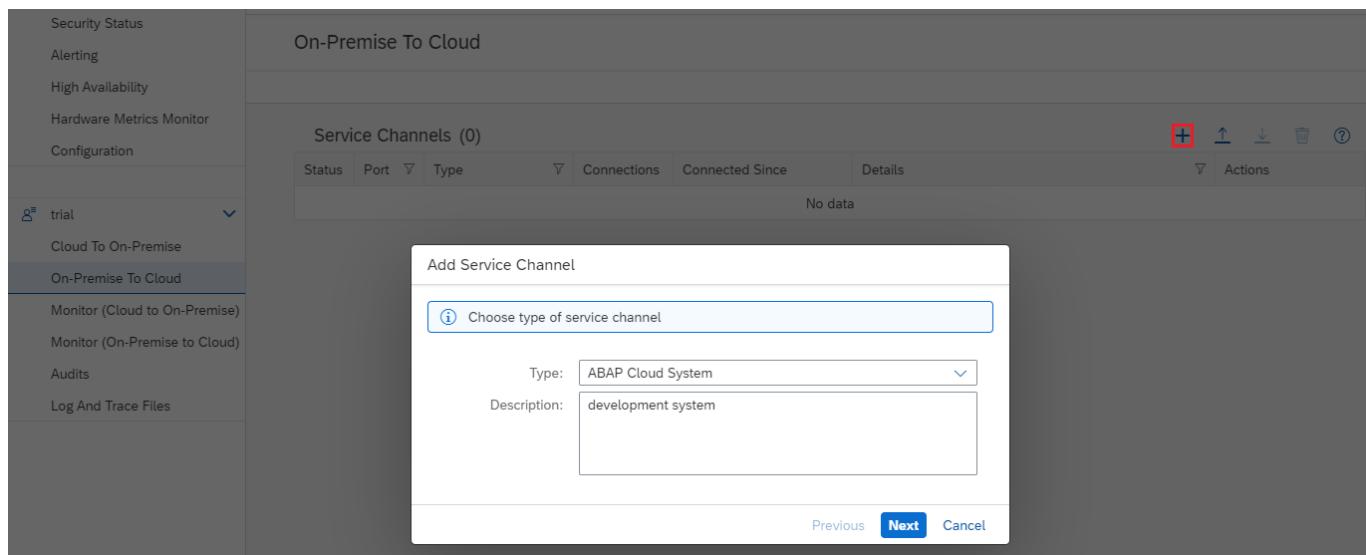
For scenarios that need to call from on-premise systems to SAP BTP ABAP environment using RFC, you can establish a connection to an ABAP Cloud tenant host. To do this, select **On-Premise to Cloud > Service Channels** in the Cloud Connector.

## 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

- From your subaccount menu, choose **On Premise To Cloud**.
- Choose the **Add (+)** icon.

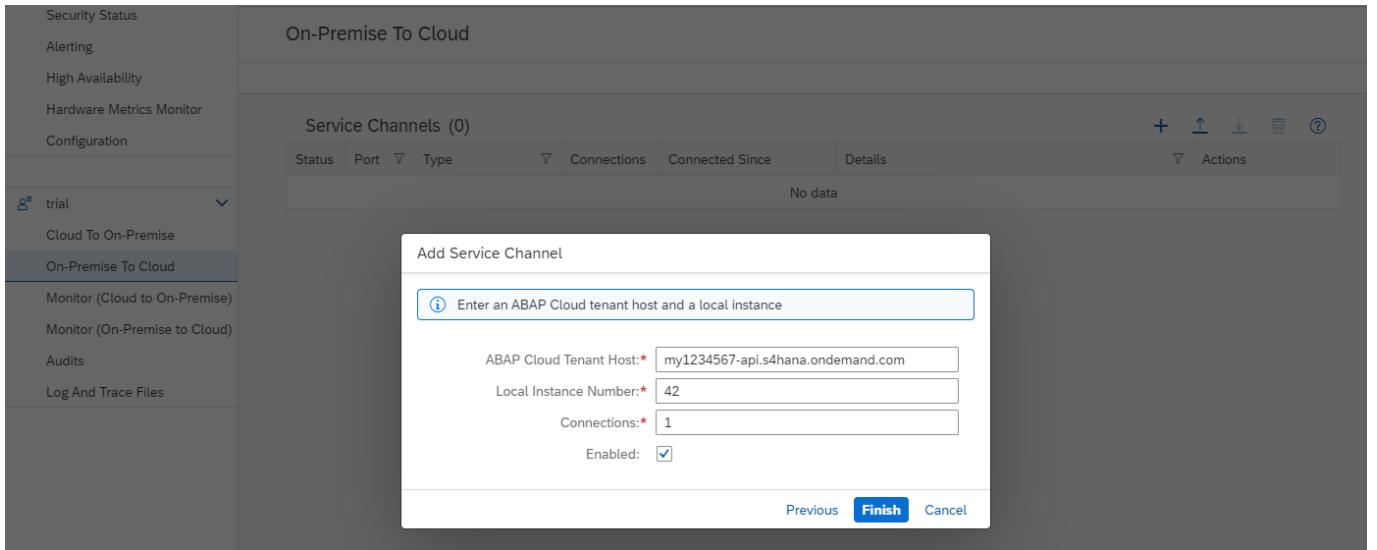


- In the **Add Service Channel** dialog, select **ABAP Cloud System** from the drop-down list of supported channel types.
- Optionally, provide a **Description** that explains what the ABAP Cloud service channel is used for.
- Choose **Next**. The **ABAP Cloud System** dialog opens.
- Enter the **<ABAP Cloud Cloud Tenant Host>** that you want to connect to.

### i 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>.
8. 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.



9. Choose **Finish**.

### i 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.

## 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: **K8s Cluster**

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: \* **kyma-proxy.region.sap**

K8s Service Host: \* **service.host**

Local Port: \* **20001**

Connections: \* **1**

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>.

## 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 (SAP HANA Database, Virtual Machine, ABAP Cloud System, Kubernetes Cluster)
- Assigned subaccount (display name)
- Details (for example, the assigned SAP HANA instance name or 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 shows the SAP Cloud Connector Administration interface. The left sidebar has a tree view with nodes like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', 'Configuration', 'trial' (selected), 'Cloud To On-Premise', 'On-Premise To Cloud', 'Monitor (Cloud to On-Premise)', 'Monitor (On-Premise to Cloud)', 'Audits', and 'Log And Trace Files'. The main content area has tabs for 'Select Subaccount' (selected) and 'Cross-Subaccount'. Under 'Select Subaccount', there's a 'Connector' section with a 'Connector Overview' table showing details: Connector ID: ABCDEFABCDEF12345678901234567890, Local Name: scc.mycompany.corp, Local IP: 10.11.12.13, Security Status: Low risk (orange), Availability: High, Disconnected, Alerts: 1. Below this is a 'Subaccount Dashboard' table with three rows: 819986f5-460d-4a36-b..., trial, and ztgif9vgj9. The final section is 'Service Channels Overview' with a table showing three service channels: 1042 (Virtual Machine, myVirtualMachine), 3332 (ABAP Cloud System, my312345-api.s4hana.ondemand.com), and 31615 (HANA Database, abcd). Filter buttons are visible above the 'Service Channels Overview' table.

Status	Port	Type	Subaccount	Details	Actions
<span style="color: green;">□</span>	1042	Virtual Machine	Virtual Machine Test	myVirtualMachine	<span style="color: blue;">🔗</span> <span style="color: blue;">✎</span> <span style="color: blue;">trash</span> <span style="color: blue;">...</span> <span style="color: blue;">🔍</span>
<span style="color: green;">□</span>	3332	ABAP Cloud System	qual	my312345-api.s4hana.ondemand.com	<span style="color: blue;">🔗</span> <span style="color: blue;">✎</span> <span style="color: blue;">trash</span> <span style="color: blue;">...</span> <span style="color: blue;">🔍</span>
<span style="color: orange;">◊</span>	31615	HANA Database	trial	abcd	<span style="color: blue;">🔗</span> <span style="color: blue;">✎</span> <span style="color: blue;">trash</span> <span style="color: blue;">...</span> <span style="color: blue;">🔍</span>

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.

# Configure Trust

Set up an allowlist for cloud applications and a trust store for on-premise systems in the Cloud Connector.

## Tasks

[Trust Cloud Applications in the Cloud Connector](#)

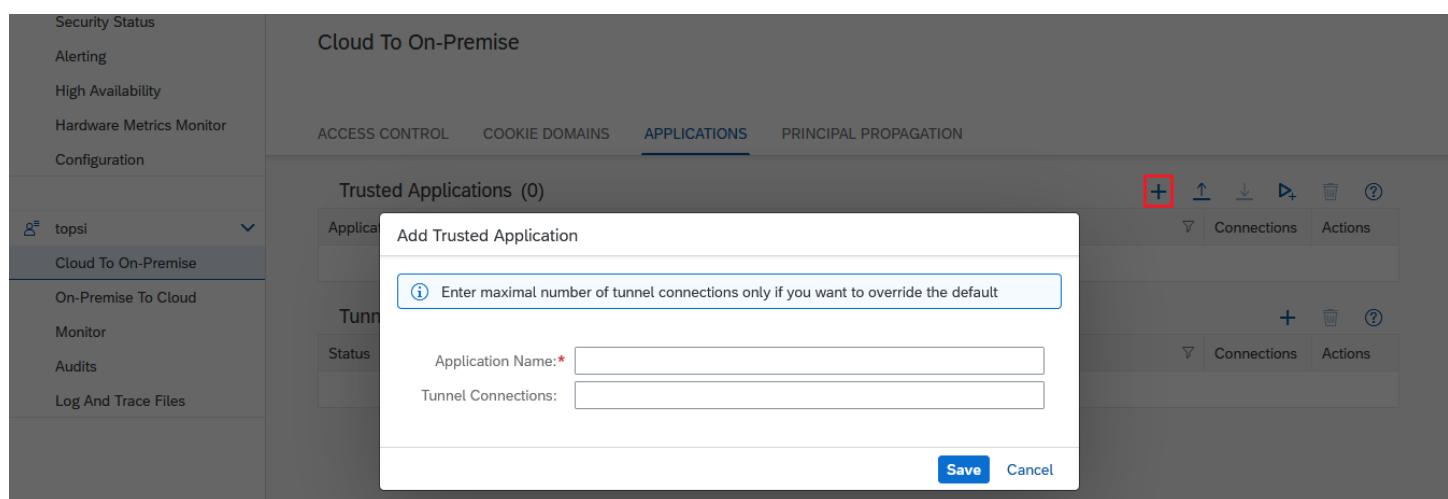
[Configure the Trust Store](#)

## Trust Cloud Applications in the Cloud Connector

### ! Restriction

Currently, the complete implementation of this feature is available only for interaction with the Neo environment.

By default, all applications within a subaccount are allowed to use the Cloud Connector associated with the subaccount they run in. However, this behavior might not be desired in specific scenarios. For example, this may be acceptable for some applications, as they must interact with on-premise resources, while other applications, for which it is not transparent whether they try to receive on-premise data, might turn out to be malicious. For such cases, you can use an application allowlist.



As long as there is no entry in this list, all applications are allowed to use the Cloud Connector. If one or more entries appear in the allowlist, then only these applications are allowed to connect to the exposed systems in the Cloud Connector.

You can add, edit, or delete entries as follows:

1. From your subaccount menu, choose **Cloud to On-Premise** and go to the **Applications** tab.
2. To add an application, choose the **Add** icon in section **Trusted Applications**.
3. Enter the *<Application Name>* in the **Add Trusted Application** dialog.

### i Note

To add all applications that are listed in section **Tunnel Connection Limits** on the same screen, you can also use the **Upload** button next to the **Add** button. The list **Tunnel Connection Limits** shows all applications for which a specific maximal number of tunnel connections was specified. See also: [Configure Advanced Connectivity](#).

4. (Optional) Enter the maximal number of *<Tunnel Connections>* only if you want to override the default value.
5. Choose **Save**.

### i Note

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

The application name is visible in the SAP BTP cockpit under **Applications > Java Applications**. To allow a subscribed application, you must add it to the allowlist in the format <providerSubaccount>: <applicationName>. In particular, when using HTML5 applications, an implicit subscription to **services:dispatcher** is required.

To edit an existing entry:

1. Choose the **Edit** button.
2. When you are done, select **Save**.

To remove an application from the list:

1. Select the entry.
2. Choose **Delete**.

To delete all entries, choose **Delete All**.

To add all applications from section **Tunnel Connection Limits** to the allowlist, choose the button **Add all applications...** from section **Trusted Applications**.

The screenshot shows the SAP BTP cockpit interface. On the left, there is a sidebar with navigation links: Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, d036325trial (selected), Cloud To On-Premise, On-Premise To Cloud, Monitor, Audits, and Log And Trace Files. The main content area has tabs: ACCESS CONTROL, COOKIE DOMAINS, APPLICATIONS (selected), and PRINCIPAL PROPAGATION. Under the APPLICATIONS tab, there are two sections: Trusted Applications (0) and Tunnel Connections Limits (1). The Trusted Applications section has a table with columns: Application Name, Connections, and Actions. A note says "White-list is empty — all applications will be trusted". The Tunnel Connections Limits section has a table with columns: Status, Application Name, Connections, and Actions. One entry is listed: test (Status: OK, Connections: 2).

Back to [Tasks](#)

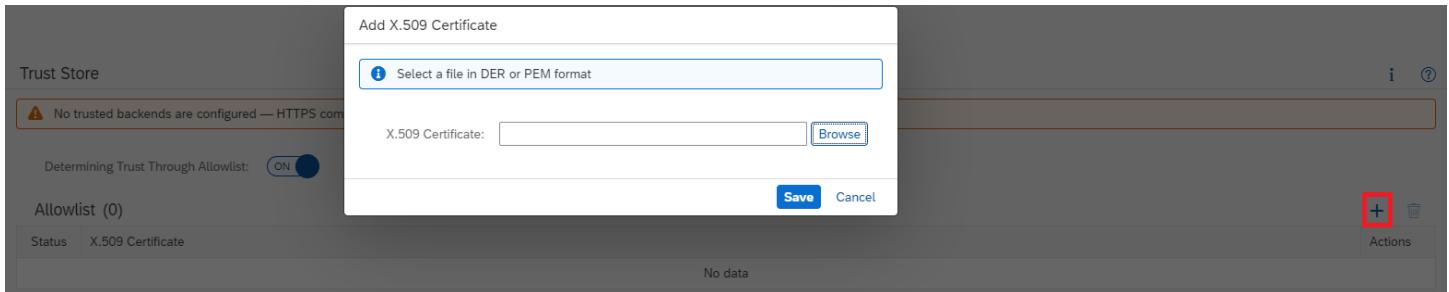
## Configure the Trust Store (as of Version 2.15)

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:

The screenshot shows the Trust Store configuration page. At the top, there is a message: "No trusted backends are configured — HTTPS communication is not possible". Below this, there is a toggle switch labeled "Determining Trust Through Allowlist" which is set to "ON". The Allowlist section shows a table with columns: Status, X.509 Certificate, and Actions. The table is empty and displays "No data".

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.



## i 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 you are going to trust, but rather **trust all backends**, you can switch off the handle. In this case, the allowlist is ignored. This option considered less secure, since all backends are trusted now.



[Back to Tasks](#)

# 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**.

Virtual Domain	Internal Domain	Actions
cloud	mycompany.corp	

The Cloud Connector checks the Web server's response for Set-Cookie headers. If it finds one with an attribute domain=intrantet.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](http://www1.sales.cloud) it attaches the cookie, which then successfully arrives at the server on machine1.intranet.corp.

### i 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.

### i 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](http://www1.sales.cloud) and replaces the cookie domain accordingly.

## Related Information

[Configure Access Control](#)

## 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. As of Cloud Connector version 2.11.2, 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.

### i Note

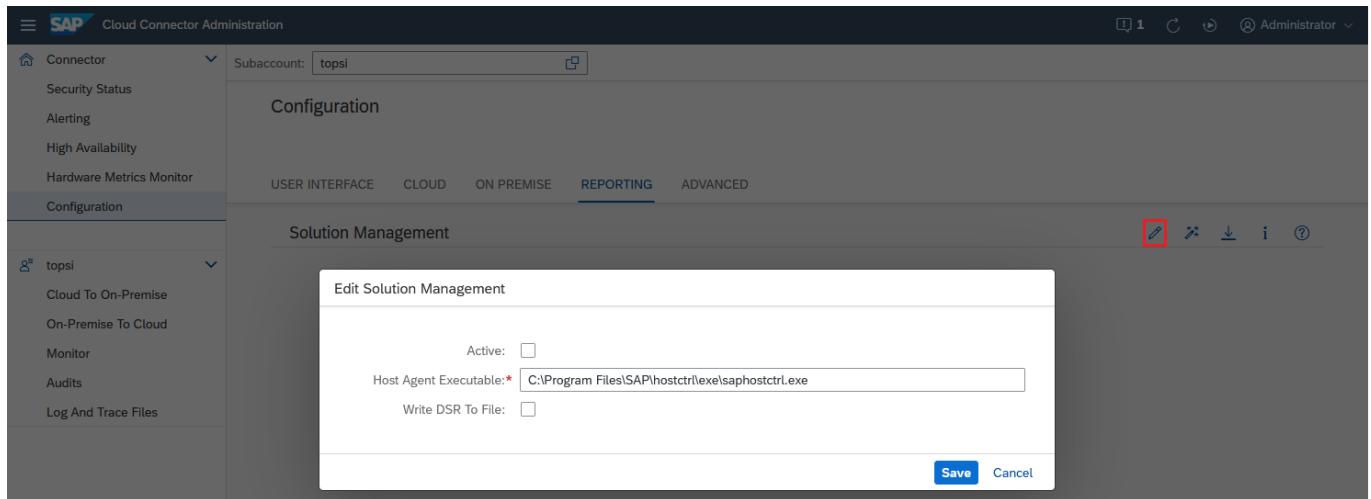
Linux OS: if you installed the host agent after installing the Cloud Connector, you can execute `enableSolMan.sh` in the installation directory (available as of Cloud Connector version 2.11.2) to adjust the host agent configuration and user group setup. This action requires root permission.

- 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**.

### i Note

To download the registration file ***ImdbModel.xml***, choose the icon **Download registration file** from the **Reporting** tab.

## Related Information

[Monitoring](#)

## 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)

### i 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](#).

To change the parameter values, do the following:

- From the Cloud Connector main menu, choose **Configuration > Advanced**. In section **Connectivity**, select **Edit**.

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar has a 'Connector' section with various sub-options like Security Status, Alerting, High Availability, Hardware Metrics Monitor, and Configuration (which is selected). Below this is a list of monitors: 15f9882b-cf6f-40db-aa7... (selected), Cloud To On-Premise, On-Premise To Cloud, Monitor (Cloud to On-Premise), Monitor (On-Premise to Cloud), and Audits. The main area is titled 'Configuration' and has tabs for USER INTERFACE, CLOUD, ON PREMISE, REPORTING, and ADVANCED (which is selected). Under ADVANCED, there's a 'Connectivity' section. It shows several parameters: Application Tunnel Connections: 2, Tunnel Worker Threads: 10, Protocol Processor Worker Threads: 20, and Max. Reconnect Attempts: 150. To the right of these are their respective max values: Max. Chunk Size HTTP Packages (kb): 64, Max. HTTP Request Header Length (kb): 64, Max. Size HTTP Request (kb): 8, Max. HTTP Response Header Length (kb): 8, and Max. Size HTTP Response (kb): 8. An 'Edit' button is located to the right of the connectivity parameters, and it is highlighted with a red box.

- In the **Edit Connectivity Settings** dialog, change the parameter values as required.

- Choose **Save**.

Additionally, you can specify the number of allowed tunnel connections for each application that you have specified as a [trusted application](#).

### i 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.

To add a specific connection limit to a trusted application, do the following:

- From your subaccount menu, choose **Cloud To On-Premise > Applications**. In section **Tunnel Connection Limits**, choose **Add**.

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar shows navigation options like Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, and a selected subaccount 'topsi' under 'Cloud To On-Premise'. The main area has tabs for ACCESS CONTROL, COOKIE DOMAINS, APPLICATIONS (which is selected), and PRINCIPAL PROPAGATION. Under APPLICATIONS, there are two sections: 'Trusted Applications (0)' and 'Tunnel Connections Limits (1)'. The 'Trusted Applications' section has a table with one row for 'abc'. The 'Tunnel Connections Limits' section has a table with one row for 'abc' with a value of 5 in the 'Connections' column.

2. In the **Edit Tunnel Connections Limit** dialog, enter the <Application Name> and change the number of <Tunnel Connections> as required.

### i Note

The application name is visible in the SAP BTP cockpit under **Applications** **Java Applications**. To allow a subscribed application, you must add it to the allowlist in the format <providerSubaccount>:<applicationName>. In particular, when using HTML5 applications, an implicit subscription to **services:dispatcher** is required.

3. Choose **Save**.

To edit this setting, select the application from the **Limits** list and choose **Edit**.

## 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

The screenshot shows the Cloud Connector configuration interface under the ADVANCED tab. The 'Connectivity' section contains several configuration parameters. A red box highlights the 'Max. Chunk Size HTTP Packages (kb)', 'Max. HTTP Request Header Length (kb)', 'Max. Size HTTP Request (kb)', 'Max. HTTP Response Header Length (kb)', and 'Max. Size HTTP Response (kb)' fields, which are all set to 64 or 8 kb.

Application Tunnel Connections:	1
Tunnel Worker Threads:	10
Protocol Processor Worker Threads:	20
Max. Reconnect Attempts:	150
Max. Chunk Size HTTP Packages (kb):	64
Max. HTTP Request Header Length (kb):	64
Max. Size HTTP Request (kb):	8
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

Parameter	Description	Default Value
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

## 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)

### i Note

A restart is required when changing JVM settings.

We recommended 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:

- From the Cloud Connector main menu, choose **Configuration > Advanced**. In section **JVM**, select **Edit**.

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar shows navigation links like Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, and Configuration. Under Configuration, there's a sub-account named 'topsi'. The main content area has a 'Configuration' title and tabs for USER INTERFACE, CLOUD, ON PREMISE, REPORTING, and ADVANCED. The ADVANCED tab is selected. Below it, there's a 'Connectivity' section with fields for Application Tunnel Connections (1), Tunnel Worker Threads (10), and Protocol Processor Worker Threads (20). Under the 'JVM' section, it shows Initial Heap Size (1024m), Maximal Heap Size (1024m), and Maximal Direct Memory (2G). A red box highlights the edit icon (pencil) next to the Maximal Direct Memory value.

- In the **Edit JVM Settings** dialog, change the parameter values as required.

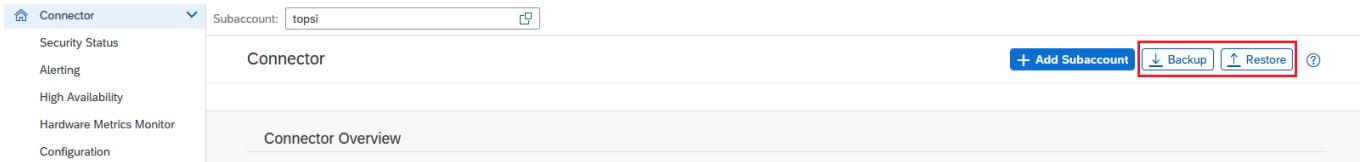
- Choose **Save**.

# Configuration Backup

You can backup and restore your Cloud Connector configuration.

To backup or restore your Cloud Connector configuration, do the following:

- 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.

- To backup your configuration, enter and repeat a password in the **Backup** dialog and choose **Backup**.

### **i** 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.

- 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.

### **i** 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.

## 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** and press the **Edit** button in section **Login Screen Information**.

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar has a 'Connector' dropdown set to 'Subaccount: trial'. The main navigation bar has tabs: USER INTERFACE (selected), CLOUD, ON PREMISE, REPORTING, and ADVANCED. Under 'USER INTERFACE', there's a sub-section titled 'Login Screen Information'. It contains several configuration fields:
 

- Display Policy: Show for both master and shadow installations
- Text Alignment: Center
- Display Width: 50%
- Background Color: #ffffff
- Display Height: auto
- Opacity (Alpha): 0.5
- Position: 10px from top
- Font Color: #2b3f7b

 Below these settings is a large text input field labeled 'Login Information:'.

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: Background Color:  Display Height: Opacity (Alpha): Text Alignment:  Font Color:  Position:   from top  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**

2. In section **Display Policy**, select a display policy for the login screen information.

3. 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 %).

**i 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.

### i 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 ( <code>left</code> , <code>right</code> , <code>center</code> , or <code>justified</code> )
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"> <li><code>href</code> (its value must be a &lt;URL&gt; with protocol <code>http</code> or <code>https</code>)</li> <li><code>target</code> (its value must be "<code>_blank</code>")</li> </ul>

HTML syntax checking is strict. Attribute values must be enclosed by double quotes. Missing or unmatched opening or closing tags are not permitted.

### i Note

Tag `br` does not require a closing tag as there cannot be any inner HTML.

## Administration

Learn more about operating the Cloud Connector, using its administration tools and optimizing its functions.

Topic	Description

Topic	Description
<a href="#">Exchange UI Certificates in the Administration UI</a>	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.
<a href="#">Configure Named Cloud Connector Users</a>	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.
<a href="#">High Availability Setup</a>	The Cloud Connector lets you install a redundant (shadow) instance, which monitors the main (master) instance.
<a href="#">Change the UI Port</a>	Use the changeport tool (Cloud Connector version 2.6.0+) to change the port for the Cloud Connector administration UI. .
<a href="#">Connect and Disconnect a Cloud Subaccount</a>	As a Cloud Connector administrator, you can connect the Cloud Connector to (and disconnect it from) the configured cloud subaccount.
<a href="#">Secure the Activation of Traffic Traces</a>	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+).
<a href="#">Monitoring</a>	Use various views to monitor the activities and state of the Cloud Connector.
<a href="#">Alerting</a>	Configure the Cloud Connector to send email alerts whenever critical situations occur that may prevent it from operating.
<a href="#">Audit Logging</a>	Use the auditor tool to view and manage audit log information (Cloud Connector version 2.2+).
<a href="#">Troubleshooting</a>	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.
<a href="#">Process Guidelines for Hybrid Scenarios</a>	How to manage a hybrid scenario, in which applications running on SAP BTP require access to on-premise systems using the Cloud Connector.
<a href="#">Configuring Backup</a>	Find an overview of backup procedures for the Cloud Connector.

## 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

- From the main menu, choose **Configuration** and go to the **User Interface** tab.

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

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.

The screenshot shows the 'Generate CSR' dialog box with the following details:

- Key Size:** 4096 Bits (selected)
- Subject DN:**
  - Common Name (CN): scc.internal.corp
  - E-Mail Address (EMAIL): (empty)
  - Locality (L): (empty)
  - Organizational Unit (OU): department
  - Organization (O): companyName
  - State or Province (ST): (empty)
  - Country (C): DE
- Subject Alternative Names:**

Type	Value	Actions
RFC822	ownerscc@company.com	<span>trash</span>
DNS	*.internal.corp	<span>trash</span>
- Buttons:** Generate (blue), Cancel

4. Press **Generate**.

5. 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.

### i Note

The response should be either an X.509 certificate or a PKCS#7 in PEM format.

6. To import the signing response, choose the **Upload** icon.

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a sidebar with 'Connector' selected. The main area has a 'Subaccount: trial' dropdown. Below it, the 'Configuration' tab is active. Under 'USER INTERFACE', the 'UI Certificate' section is visible. In the top right of this section, there's a toolbar with several icons, one of which is highlighted with a red box.

As of Cloud Connector version 2.13, you can also upload an existing PKCS#12 certificate directly (instead of generating a CSR).

This screenshot shows the same SAP Cloud Connector Administration interface as above, but with a modal dialog open over the 'UI Certificate' section. The dialog is titled 'Import UI Certificate'. It contains fields for 'Subject DN', 'Issuer', 'Valid From', and 'Valid To'. Below these, there are fields for 'Certificate Type' (radio buttons for 'Signed Certificate' and 'P12 Certificate', with 'P12 Certificate' selected), 'P12 Certificate' (with a 'Browse' button), and 'Password'. At the bottom right of the dialog, there are 'Import' and 'Cancel' buttons, both of which are highlighted with red boxes.

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.

## 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`

**i Note**

The role `sccmonitoring` provides access to the monitoring APIs, and is particularly used by the SAP Solution Manager infrastructure, see [Monitoring APIs](#).

- Support users: `sccsupport`

Alternatively, you can define custom role names for each of these user groups, see: [Use LDAP for Authentication](#).

Once configured, the default Cloud Connector Administrator user becomes inactive and can no longer be used to log on to the Cloud Connector.

## Use LDAP for Authentication

You can use LDAP (Lightweight Directory Access Protocol) to configure Cloud Connector 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:

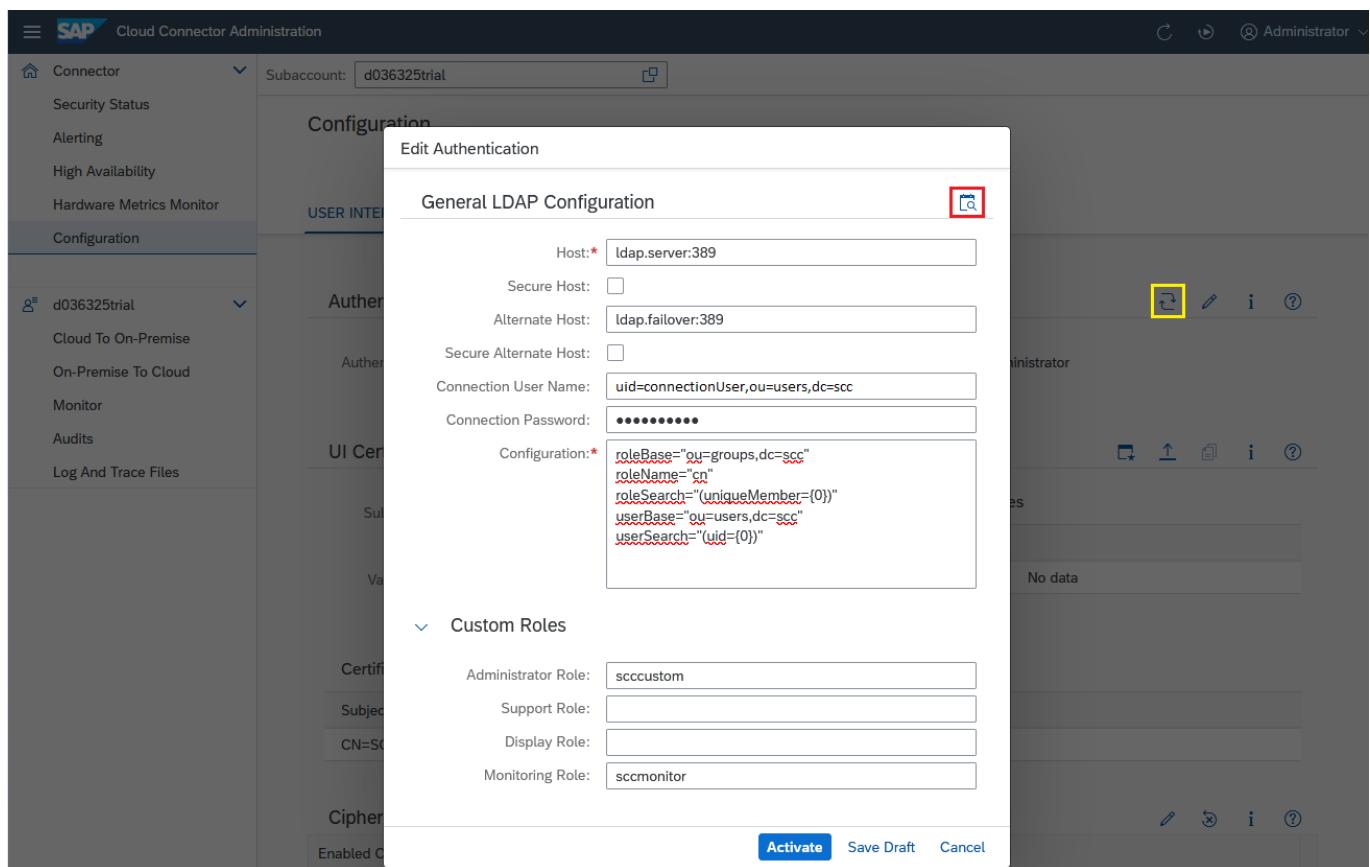
User Role	Authorization
<code>sccadmin</code> or <code>admin</code>	Administrates the Cloud Connector (all CRUD operations).
<code>sccsubadmin</code>	<ul style="list-style-type: none"> <li>• Manage all subaccount-related settings.</li> <li>• Perform support-related tasks like setting trace levels or creating a thread dump.</li> </ul> <p>Access to common settings for all subaccounts, like system certificate settings, is permitted.</p>
<code>sccdisplay</code>	Access the Cloud Connector administration UI in read-only mode.

User Role	Authorization
sccsupport	<ul style="list-style-type: none"> <li>Access the Cloud Connector administration UI in read-only mode.</li> <li>Perform support-related tasks like setting trace levels or creating a thread dump.</li> </ul>
sccmonitoring	Provides access to the monitoring APIs, and is particularly used by the SAP Solution Manager infrastructure, see <a href="#">Monitoring APIs</a> .

Group membership is checked by the Cloud Connector.

## Setting LDAP Authentication

- From the main menu, choose **Configuration** and go to the **User Interface** tab.
- From the **Authentication** section, choose **Switch to LDAP**.



- (Optional) To save intermediate adoptions of the LDAP configuration, choose **Save Draft**. This lets you store the changes in the Cloud Connector without activation.
- 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.

## i 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.

## i 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](https://tomcat.apache.org/tomcat-8.5-doc/realm-howto.html).

## i 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](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 **Authentication** section.

## i 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 SSL/TLS can cause SSL 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 cacert
```

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 again.

### i 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"
           <Realm X509UsernameRetrieverClassName="com.sap.scc.tomcat.utils.SccX509SubjectDnRetriever
         </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

Get background information on LDAP configuration for the Cloud Connector.

[Introduction](#)

[Connect to the LDAP Server](#)

[SSL Issues](#)

[Authentication](#)

[User Selection](#)

[User Roles](#)

[Relationship between User and Group](#)

[Custom User Roles](#)

[Additional Notes](#)

## 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/realm-howto.html](http://tomcat.apache.org/tomcat-8.5-doc/realm-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](#)

## 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 SSL/TLS) or plain. It points to a host and port. The address looks like this:

- SSL/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 SAP Cloud Connector interface for editing authentication. On the left, under 'Edit Authentication', there is a red error box containing the message 'LDAP server authentication failed' with a link to 'Details'. Below this, the 'General LDAP Configuration' section is visible, showing fields for Host, Secure Host, Alternate Host, Secure Alternate Host, Connection User Name, Connection Password, and Configuration (set to 'x="x"'). On the right, a 'Test Results' panel displays the following information:

- Host:** Reachable
- Authentication:** Failed
- Details:**
  - Connecting to URL `ldap://wdflbmd15468.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://wdflbmd15468.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
- What You Can Do:** Verify connection user and password
- For Support Purposes:** [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](#)

## SSL Issues

LDAP connection over SSL/TLS will run into SSL 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
```

See also: [Working with Certificates and SSL](#) (Java documentation).

Back to [Top](#)

## 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.

### i 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"
```

### i 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=
```

You can perform the same test in the Cloud Connector UI:

Edit Authentication

### General LDAP Configuration

*Host:	<input type="text" value="10.10.10.10:10389"/>
Secure Host:	<input type="checkbox"/>
Alternate Host:	<input type="text"/>
Secure Alternate Host:	<input type="checkbox"/>
Connection User Name:	<input type="text" value="uid=admin,ou=system"/>
Connection Password:	<input type="password" value="****"/>
*Configuration:	<input type="text" value="x=x"/>

### Custom Roles

Administrator Role:	<input type="text"/>
Support Role:	<input type="text"/>
Display Role:	<input type="text"/>
Monitoring Role:	<input type="text"/>

Save Draft   Activate   Cancel

Choose the test icon in the upper right corner, and enter something as name and password:

Cloud Connector Administration

Edit Authentication

Authentication failed →[Details](#)

General LDAP Configuration

*Host:	<input type="text" value="10.10.10.10:10389"/>
Secure Host:	<input type="checkbox"/>
Alternate Host:	<input type="text"/>
Secure Alternate Host:	<input type="checkbox"/>
Connection User Name:	<input type="text" value="uid=admin,ou=system"/>
Connection Password:	<input type="password" value="****"/>

Test Results

Host: Reachable

Authentication: Failed

Details:

- Connecting to URL ldap://10.10.10.10:10389
- No user found.
- Authentication for user ttt failed

What You Can Do:

Verify LDAP user configuration and user name (for login)

For Support Purposes:

[Download test results including LDAP configuration](#)

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](#)

## 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"
```

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:

**Edit Authentication**

**No roles assigned to user Thor →[Details](#)**

**G Test Results**

Host:  
Reachable

Authentication:  
Succeeded

Details:

- Connecting to URL ldap://...
- 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](#)

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.

**Test Results**

Host:  
Reachable

Authentication:  
Succeeded

Details:

- Connecting to URL ldap://[REDACTED]
- Search returned relative name: uid=Thor,ou=users
- 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](#)

No data

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})"`.

### i 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" userSearch="(cn=Thor)"
```

### i 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](#)

## User Roles

At this point, the configuration let's 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>" "({
```

The screenshot shows the SAP Cloud Connector configuration interface. On the left, under 'Edit Authentication', there is a red error box stating 'No roles assigned to user Thor →Details'. Below it, 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"). On the right, the 'Test Results' panel shows the host is reachable and authentication succeeded. It details the search process for user Thor and lists found roles. A link to download test results is provided.

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](#)

## 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 shows a SAP Fiori application interface. On the left, there's a configuration form titled "Edit Authentication" under "ADVANCED". It includes fields for "Host", "Secure Host", "Alternate Host", "Secure Alternate Host", "Connection User Name" (set to "uid=admin,ou=system"), "Connection Password" (redacted), and "Configuration" which contains LDAP search parameters. A red box highlights an error message: "No roles assigned to user Thor →Details". Below this is a section for "Custom Roles". At the bottom are buttons for "Activate", "Save Draft", and "Cancel".

**Test Results**

Host: Reachable  
Authentication: Succeeded

Details:

- Connecting to URL `ldap://wdflbmd15468.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)

What You Can Do:  
Verify LDAP configuration and custom roles

For Support Purposes:  
[Download test results including LDAP configuration](#)

User: Administrator

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`:

The screenshot shows the SAP Fiori interface for editing authentication settings. On the left, under 'General LDAP Configuration', fields include Host (IP address), 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", roleSearch="(uniqueMember={0})"). A green box highlights 'User Thor authenticated' with a link to 'Details'. On the right, the 'Test Results' section shows 'Active Roles: sccadmin' and a detailed log of the authentication process.

**Test Results**

Active Roles:  
sccadmin

Details:

- Connecting to URL ldap://wdflbmd15468.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](#)

### i 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](#)

## 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.

### i 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](#)

## 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](#)

# 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
<a href="#">Install a Failover Instance for High Availability</a>	Install a redundant Cloud Connector instance (shadow) that monitors the main instance (master).
<a href="#">Master and Shadow Administration</a>	Learn how to operate master and shadow instances.

## Related Information

[Connect DB Tools to SAP HANA via Service Channels](#)

## 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.

### i Note

For detailed information about sizing of the master and the shadow instance, see also [Sizing Recommendations](#).

## 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**.

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a navigation sidebar with sections like 'Connector', 'Security Status', 'Alerting', 'High Availability' (which is selected and highlighted in blue), 'Hardware Metrics Monitor', and 'Configuration'. Below that is another sidebar for 'd036325trial' with options like 'Cloud To On-Premise', 'On-Premise To Cloud', 'Monitor', 'Audits', and 'Log And Trace Files'. The main content area has a header 'High Availability' with a status 'Disabled'. It includes sections for 'Connection To Shadow' (with fields for 'Shadow Host' and 'Shadow Port') and 'Shadow Component Versions' (with fields for 'Tunnel', 'SCC', 'LJS', and 'JRE', all marked as 'n.a.'). At the top right, there are 'Enable', 'Reset', and '...' buttons, with 'Enable' being the one highlighted with a red box.

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.

### **i 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**:

The screenshot shows a UI wizard window with the title 'Choose Installation Type'. It contains two radio button options: 'Master (Primary Installation)' and 'Shadow (Backup Installation)', with the latter being selected. At the bottom right of the window is a blue 'Save' button with a white icon.

2. From the main menu, choose **Shadow Connector** and provide connection data for the master instance, that is, the master host and port. As of version 2.8.1.1, 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.

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a sidebar with options like 'Shadow Configuration', 'Mirrored Master Configuration', 'Log And Trace Files', 'Mirrored Subaccounts' (selected), 'Cloud To On-Premise', and 'On-Premise To Cloud'. The main area is titled 'Shadow Connector' and shows a status message 'Disconnected'. Below it, a sub-section 'Connection To Master' is open, displaying the 'Edit Connection To Master' dialog. This dialog contains fields for 'Master Host' (sccmaster.mycompany.org), 'Master Port' (8443), 'Own Host' (sccshadow.mycompany.org), 'Check Interval' (60), and 'Takeover Delay' (30). At the bottom of the dialog are 'Save' and 'Cancel' buttons.

### i 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.

3. 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.
4. The UI on the master instance shows information about the connected shadow instance. From the main menu, choose **High Availability**:

The screenshot shows the SAP Cloud Connector Administration interface with the 'High Availability' view selected in the sidebar. The main area displays a message 'Connected since 29 October 2020 17:31:59 CET'. Below this, there's a 'Connection To Shadow' section with details: 'Shadow Host: sccshadow.mycompany.org', 'Shadow Port: 8555', and 'Most Recent Check: 29 October 2020 17:31:59 CET'. At the bottom, there's a 'Shadow Component Versions' section showing 'Tunnel: 2.231.0', 'SCC: 2.13.0 (dev)', 'JLS: 3.118.2', and 'JRE: 1.8.0\_221 (SAP AG, C:\Program Files\Java\sapjvm\_8\jre)'.

5. As of version 2.6.0, the **High Availability** view includes an **Alert Messages** panel. It displays alerts 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. This panel lets an administrator know 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. As of version 2.8.0.1, these alerts have been integrated in the general **Alerting** section; there is no longer a separate **Alert Messages** panel.

If the master doesn't recover automatically, disconnect, then reconnect the shadow, which triggers a complete configuration transfer.

## Related Information

# 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

(i) Validate and/or change current timeout settings

<b>Timeout Settings</b>	<span style="font-size: 2em; color: #0070C0;">🔍</span>
Connection Timeout (in ms):	<input type="text" value="1000"/>
Request Timeout (in ms):	<input type="text" value="2000"/>
<b>Timeout Validation</b>	
Number of requests:	<input type="text" value="10"/>
Requests:	<span style="font-size: 1.5em;">▶</span> <span style="font-size: 1.5em; margin-left: 10px;">🔍</span>
Errors:	<span style="font-size: 1.5em;">▶</span> <span style="font-size: 1.5em; margin-left: 10px;">🔍</span>
<span style="background-color: #0070C0; color: white; padding: 5px 10px; border-radius: 5px; border: none; cursor: pointer;">Save</span> <span style="border: 1px solid #ccc; padding: 2px 10px; margin-left: 10px;">Cancel</span>	

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 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, an 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.

### i 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.

### i 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.

## 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.

#### **i 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:
  - o Microsoft Windows OS:  
`changeport <desired_port>`
  - o 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.

## 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. On the left, there's a navigation sidebar with sections like Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, and a trial section. The main area is titled 'Connector Overview' and displays connector details: ID F98F5E60FE3211E4AD44EC790A124F3F, Local Name: '.....', Local IP: '.....', Security Status: Low risk (orange), Availability: High (green), and Alerts: 0. Below this is the 'Subaccount Dashboard' table:

Status	Subaccount	Display Name	Location ID	Region	Actions
	trial	trial		Europe (Rot)	
	maasdemo	maasdemo		Europe (Rot)	

### i 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](#).

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](#)

## 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, which is supported by the Cloud Connector release **1.3.2** and higher.

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.

## i 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 use 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](#) flag.
3. To activate the payload trace, first the owner of `writeHexDump` must change the file content from `allowed=false` to `allowed=true`. Thereafter, the [Administrator](#) user can activate the payload trace from the Cloud Connector administration screens.

## Four-Eyes Principle for Linux OS/Mac OS X

1. Create a file named `writeHexDump` in `/usr/local/v1/base/cfg` (Cloud Connector 1.3.2) or `/opt/sap/scc/scc_config` (Cloud Connector 2.x). The owner of this file must be a user other than the `scctunnel` user (that is, the operating system user under which the *cloud connector* processes run) 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 set the trace level higher than `Runtime` (Cloud Connector 1.3.2) or to activate the [Payload Trace](#) flag (Cloud Connector 2.x).
3. To set a higher trace level, which includes traffic Hex-dumps (Cloud Connector 1.3.2), or to activate the payload trace (Cloud Connector 2.x), the file owner mentioned above must first change the file content from `allowed=false` to `allowed=true`. Thereafter, the [Administrator](#) user can activate one of the higher trace levels (Cloud Connector 1.3.2) or the payload trace (Cloud Connector 2.x) from the Cloud Connector administration screens.

## 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](#).

## 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](#).

The screenshot shows the SAP BTP Cockpit interface. The left sidebar has a navigation menu with items like Virtual Machines, SAP HANA / SAP ASE, Monitoring, Connectivity, Destinations, Cloud Connectors (which is selected and highlighted in blue), Security, Trust, Authorizations, OAuth, Repositories, and Integration Tokens. The main content area shows the Subaccount: FM dashboard. At the top, there's a green button labeled "Connected". Below it, the "Master instance" section provides detailed information:

Description:	my SCC
Connector Id:	B93D2570494911E4B
Connected since:	14.09.2020 12:18:22
Connected by:	here
Location Id:	here
Version:	2.12.0
Java Version:	1.7.0_55
High Availability:	active

Below that is the "Shadow instance" section:

Version:	2.12.0
Java Version:	1.7.0_55

## 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](#), whereas subaccount-specific performance and usage data is available via [Subaccount-Specific Monitoring](#). To provide external monitoring tools, you can use the [Monitoring APIs](#).

## Related Information

[Configure Solution Management Integration](#)

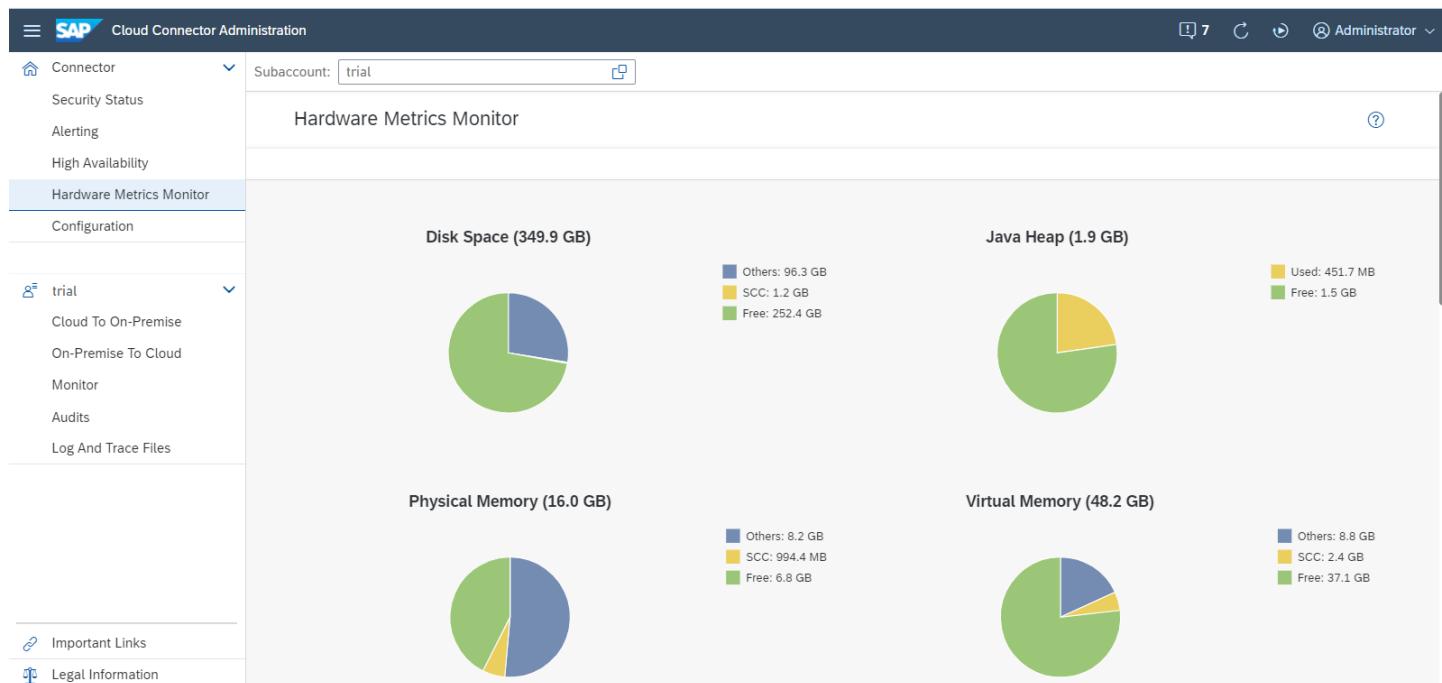
[High Availability Setup](#)

## 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.

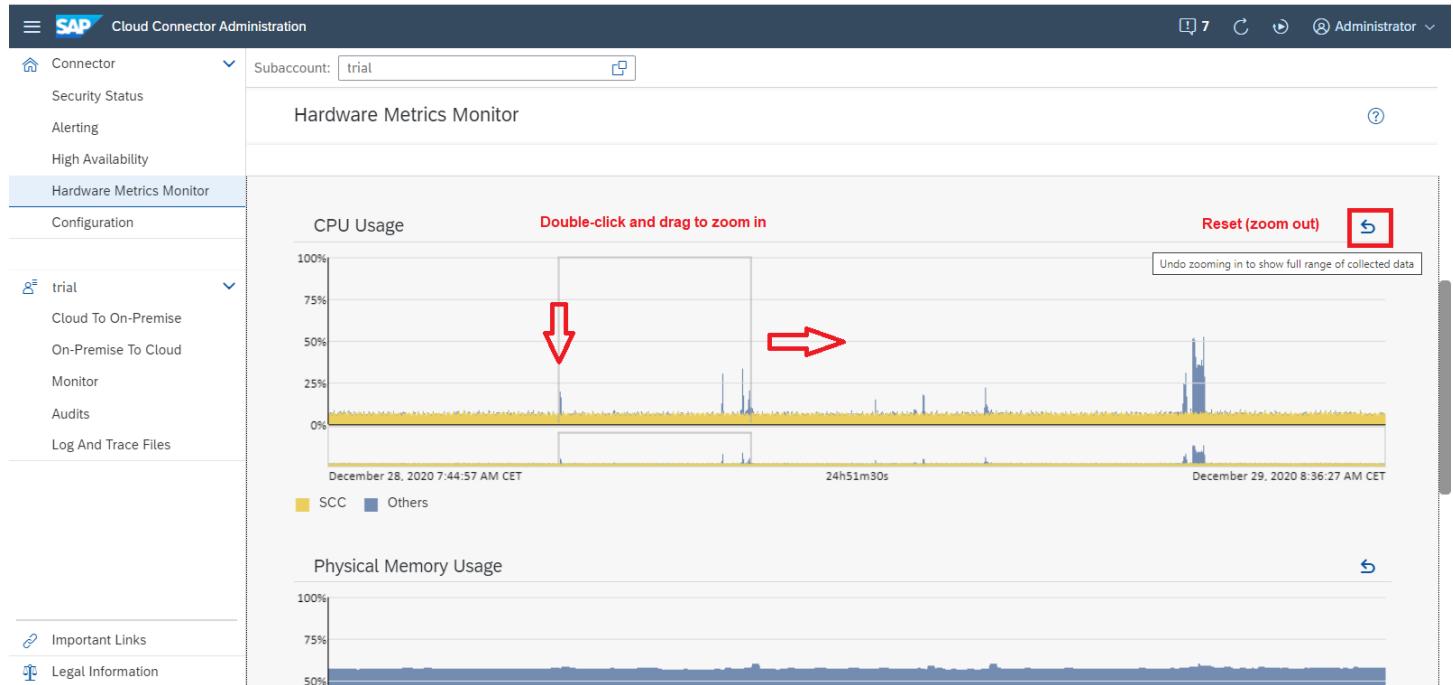
To access the monitor, choose **Hardware Metrics Monitor** from the main menu.



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).

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.



## 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 \(Cloud to On-Premise\)](#)

[Monitoring \(On-Premise to Cloud\)](#)

## Monitoring (Cloud to On-Premise)

Monitor cloud to on-premise connections in the Cloud Connector.

### Content

[Performance Overview](#)

[Most Recent Requests](#)

[Resource Filter Settings](#)

[Top Time Consumers](#)

[Usage Statistics](#)

## 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](#).

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.

### **i** Note

**Delete All** deletes not only the list of most recent requests, but it also clears the top time consumers.

Back to [Content](#)

## Most Recent Requests

This option shows the most recent requests:

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar has sections for Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, and trial (which is expanded). Under trial, there are sub-sections for Cloud To On-Premise, On-Premise To Cloud, and Monitor (which is selected). The main area has a Subaccount dropdown set to trial. The 'Monitor' section displays performance statistics and a table of most recent requests. The table has columns for Date, Virtual Host, Resource, Protocol, Duration, and Actions. Two rows are visible: one for Dec 21, 2020 3:14:34 PM to abapserver.hana.cloud:<port> with resource RFC\_RAISE\_ERROR and protocol RFC; another for Dec 21, 2020 3:14:34 PM to abapserver.hana.cloud:<port> with resource RFCPING and protocol RFC. A red box highlights the 'Select Host' dropdown in the table header, which is set to '— All Hosts —'. Another red box highlights the date column of the first request row.

Date	Virtual Host	Resource	Protocol	Duration	Actions
Dec 21, 2020 3:14:34 PM	abapserver.hana.cloud:<port>	RFC_RAISE_ERROR	RFC	68ms	
Dec 21, 2020 3:14:34 PM	abapserver.hana.cloud:<port>	RFCPING	RFC	42ms	

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  
Protocol: RFC  
Virtual Host: abapserver.hana.cloud:<port>  
Internal Host: <host>:<port>

Resource: RFCPING  
User: JCOTEST  
Bytes Sent: 304  
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.

**i Note**

Parts 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.

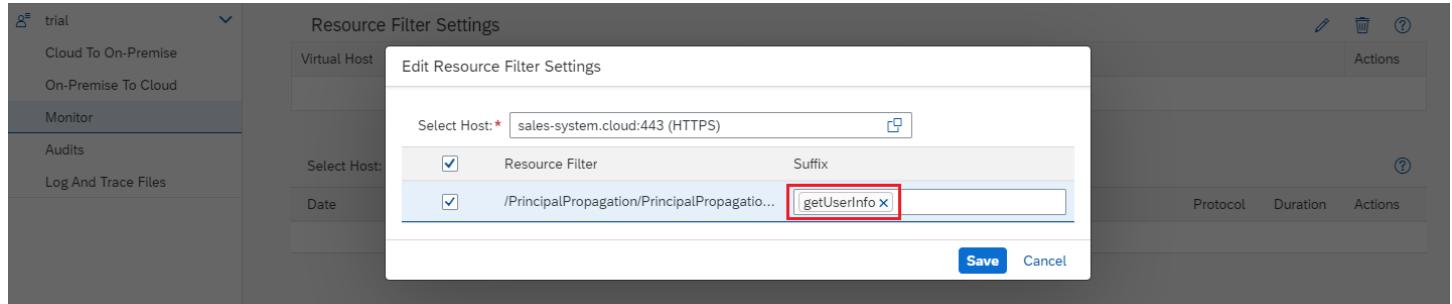
[Back to Content](#)

## 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:

Virtual Host	Exact Name or Path	Resource Filter	Actions
sales-system.cloud:443	<input checked="" type="checkbox"/>	/PrincipalPropagation/PrincipalPropagationServlet/getUserInfo	<a href="#"></a> <a href="#"></a>

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](#)). 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.

### i 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](#)

## 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](#)

## 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.

### i Note

Currently, this feature is only available for the protocols HTTP(S) and RFC (SNC). Virtual hosts using other protocols are not listed.

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a sidebar with navigation links like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', 'Configuration', and 'Audit'. Below that is a dropdown for 'Subaccount' set to 'trial'. The main area is titled 'Monitor' and displays 'Usage statistics are being collected since December 22, 2020 2:27:41 PM CET'. It has tabs for 'PERFORMANCE', 'MOST RECENT REQUESTS', 'TOP TIME CONSUMERS', 'USAGE STATISTICS' (which is selected), and 'BACK-END CONNECTIONS'. Under 'USAGE STATISTICS', there are two tables: 'Virtual Systems Usage (1)' and 'Resources Usage Of ..... (1)'. The first table shows a single entry for an RFC connection with 'Bytes Received' at 21.4 KB and 'Bytes Sent' at 12.8 KB. The second table shows a single entry for 'STFC\_CONNECTION' with 'Bytes Received' at 6190 Bytes and 'Bytes Sent' at 4803 Bytes. At the bottom, it says '24H Throughput Overview For ldcibco.wdf.sap.corp:sa...' with a data collection period from 'December 26, 2020 4:00:00 PM CET' to 'December 27, 2020 3:09:55 PM CET'. A chart titled 'Bytes Received' shows a flat line at approximately 4.5k.

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. This might be helpful when monitoring a specific use case.

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.

This screenshot shows a table titled 'Virtual Systems Usage (2)'. It has columns for 'Prot...', 'Virtual Host', 'Bytes Received', 'Bytes Sent', and 'Most'. There is a 'Filter' button at the top right of the table. A red box highlights this button. Below the table, a tooltip says: 'Show only those systems that have been used, but have unused resources'.

[Back to Content](#)

## 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](#)

## 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).

## 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.

## i Note

This API set is designed particularly for monitoring the Cloud Connector via the SAP Solution Manager, see [Configure Solution Management Integration](#).

Before you start using these APIs, please also read the general introduction to [REST APIs](#) 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.

## i 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](#) (available as of version 2.6.0)
- [Subaccount Data](#) (as of 2.10.0)
- [Open Connections to On-Premise Backend Systems](#) (as of 2.10.0)
- [Open Connections to Cloud Services](#) (as of 2.15.0)
- [Performance Data](#) (as of 2.10.0)
- [Top Time Consumers](#) (as of 2.11.0)
- [Memory Status](#) (as of 2.13.0)
- [Certificate Status](#) (as of 2.13.0)
- [Certificate Selection List](#) (as of 2.13.0)
- [Usage Statistics](#) (as of 2.13.0)
- [Master Role Check](#) (as of 2.15.0)

### Health Check (available as of version 2.6.0)

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:

<b>URI</b>	/exposed? action=ping
<b>Method</b>	GET
<b>Request</b>	
<b>Response</b>	

<b>Errors</b>	
<b>Roles</b>	All roles are accepted

Back to [Available APIs](#)

Back to [Context](#)

### List of Subaccounts (available as of version 2.10.0)

#### **i 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:

<b>URI</b>	/api/monitoring/subaccounts
<b>Method</b>	GET
<b>Request</b>	
<b>Response</b>	{subaccounts, version}
<b>Errors</b>	
<b>Roles</b>	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
  - tunnel: array of connection tunnels used by the subaccount
    - state: connected or disconnected
    - connectedSince: connection start time
    - 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)
  - recoveryAccountState: state and more details about the disaster recovery subaccount, if configured
    - isActive: disaster recovery subaccount is working (true or false)
- version: API version.

Example:

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<sccport>/api/mo
```

**Example:**

```

{
  "subaccounts": [
    {
      "tunnel": {
        "state": "Connected",
        "connectedSince": "2017-05-11T15:51:14.915 +0200",
        "connections": 0,
        "applicationConnections": [],
        "serviceChannels": [
          {
            "type": "VirtualMachine",
            "state": "ConnectFailure",
            "details": "dexlab"
          }
        ],
        "regionHost": "hana.ondemand.com",
        "subaccount": "a117",
        "locationID": ""
      },
      {
        "recoveryAccountState": {
          "isActive": false,
          "tunnel": {
            "state": "Connected",
            "connectedSince": "2017-05-12T06:58:26.613 +0200",
            "connections": 0,
            "applicationConnections": [],
            "serviceChannels": []
          },
          "regionHost": "sapbydesign.com",
          "subaccount": "dev",
          "locationID": ""
        }
      }
    }
  ]
}

```

Back to [Available APIs](#)

Back to [Context](#)

#### List of Open Connections to On-Premise Backend Systems (available as of version 2.10.0)

##### **i 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	/api/monitoring/connections/backends
Method	GET
Request	

<b>Response</b>	{subaccounts, version}
<b>Errors</b>	
<b>Roles</b>	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>:<sccport>/api,
```

**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](#)

[Back to Context](#)

#### List of Open Connections to Cloud Services (available as of version 2.15.0)

##### **i 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>:<sccport>/api,
```

[Back to Available APIs](#)

[Back to Context](#)

### Performance Monitor Data (available as of version 2.10.0)

#### i 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:

<b>URI</b>	/api/monitoring/performance/backends
<b>Method</b>	GET
<b>Request</b>	
<b>Response</b>	{subaccounts, version}
<b>Errors</b>	
<b>Roles</b>	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>:<sccport>/api/
```

*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](#)

## Top Time Consumers (available as of version 2.11.0)

### **i 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	/api/monitoring/performance/topTimeConsumers
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
  - 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>:<sccport>/api/
```

*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
        },
        ...
      ],
      "regionHost": "int.sap.hana.ondemand.com",
      "subaccount": "d039407sapdev",
      "locationID": "location"
    }
  ],
  "version": 1
}

```

Back to [Available APIs](#)

Back to [Context](#)

**Memory Status (available as of version 2.13.0)**

### i 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	/api/monitoring/memory
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**: the total size of the physical memory
  - **CloudConnector**: the size of the physical memory used by the Cloud Connector
  - **others**: the size of the physical memory used by all other processes
  - **free**: the size of the free physical memory
- **virtualKB** : usage of the virtual memory, split into four categories (all sizes in KB)
  - **total**: the total size of the virtual memory
  - **CloudConnector**: the size of the virtual memory used by the Cloud Connector
  - **others**: the size of the virtual memory used by all other processes
  - **free**: the size of the free virtual memory
- **cloudConnectorHeapKB** : usage of the Java heap, split into three categories (all sizes in KB):
  - **total**: the total size of the Java heap
  - **used**: the size of the Java heap used by the Cloud Connector
  - **free**: the size of the free Java heap

**Sample Code**

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<sccport>/api...
```

**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](#)

## Certificate Status (available as of version 2.13.0)

### **i 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	/api/monitoring/certificates
Method	GET
Request	
Response	{expired, expiring, ok}
Errors	
Roles	Administrator, Monitoring

### Response Properties:

- **expired**: the list of all expired certificates
- **expiring**: 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**: 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

A certificate in any of those lists is represented by a JSON object with the following properties:

- **type**: the 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**: the end date of the respective certificate's validity (as a long integer, that is, a UTC timestamp)
- **subjectDN**: the subject DN of the respective certificate (included only for non-subaccount certificates)
- **subaccountName**: the name of the subaccount (only for subaccount certificates)
- **subaccountRegion**: the region or landscape host of the subaccount (only for subaccount certificates)
- **isDisasterRecoverySubaccount**: a flag (that is, a Boolean value) that indicates that the subaccount is employed for disaster recovery. It can therefore be present only for subaccount certificates. Moreover, it is added only for disaster recovery subaccounts with its value set to true

### Sample Code

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<sccport>/api/
```

*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](#)

Back to [Context](#)

#### Certificate Selection List (available as of version 2.13.0)

##### **i 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	/api/monitoring/certificates/{selection}
Method	GET
Request	
Response	[Array of certificates]
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

- o 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: the 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: the end date of the respective certificate's validity (as a long integer, that is, a UTC timestamp)
- subjectDN: the subject DN of the respective certificate (included only for non-subaccount certificates)
- subaccountId: the name of the subaccount (only for subaccount certificates)
- subaccountRegion: the region or landscape host of the subaccount (only for subaccount certificates)
- isDisasterRecoverySubaccount: a flag (that is, a Boolean value) that indicates that the subaccount is employed for disaster recovery. It can therefore be present only for subaccount certificates. Moreover, it is added only for disaster recovery subaccounts with its value set to true

#### **Sample Code**

```
curl -k -H 'Accept:application/json' -u <user>:<password> -X GET https://<scchost>:<sccport>/api...
```

Back to [Available APIs](#)

Back to [Context](#)

#### **Usage Statistics (available as of version 2.13.0)**

##### **i 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:

<b>URI</b>	/api/monitoring/usage
<b>Method</b>	GET
<b>Request</b>	
<b>Response</b>	{subaccounts, version}
<b>Errors</b>	
<b>Roles</b>	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;

**i 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;

**i 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>:<sccport>/api,
```

*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](#)[Back to Context](#)

### 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.

<b>URI</b>	/exposed? action=hasMasterRole
<b>Method</b>	GET
<b>Request</b>	
<b>Response</b>	{true, false}
<b>Errors</b>	
<b>Roles</b>	All roles are accepted

[Back to Available APIs](#)[Back to Context](#)

## 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**).

### i Note

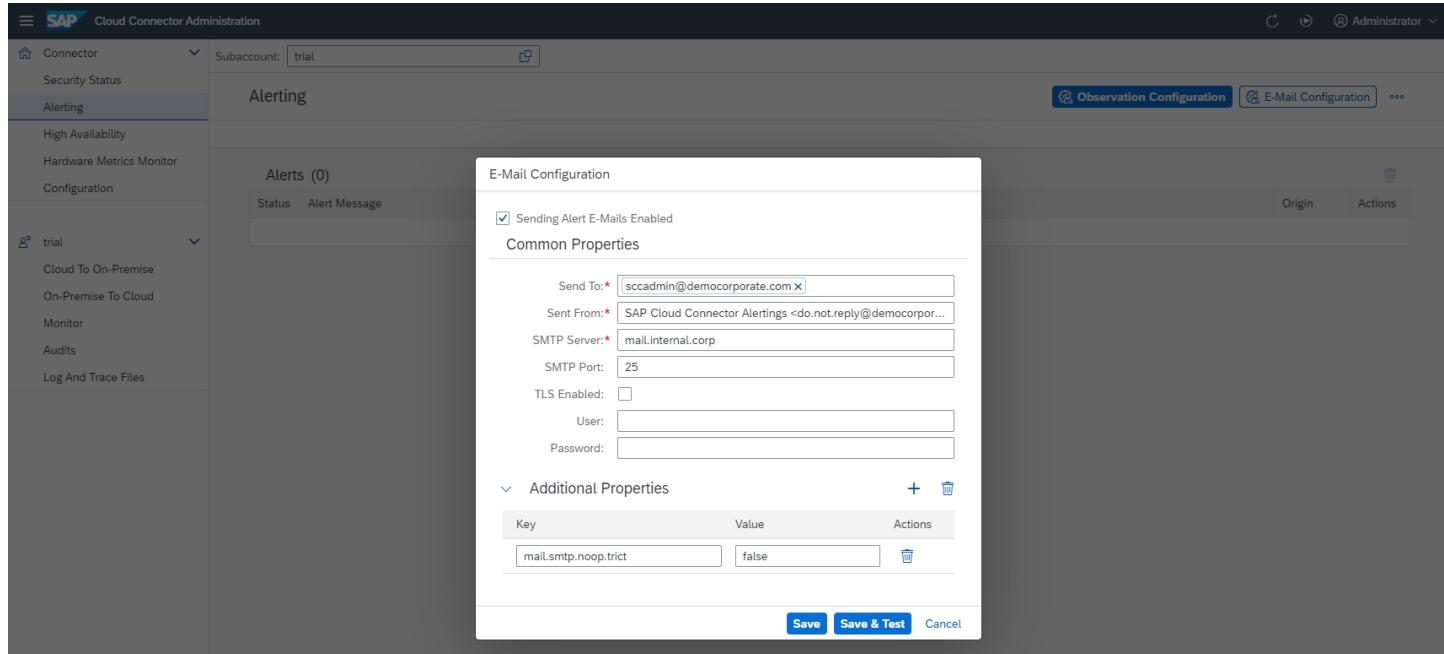
The addresses you enter here can use either of the following formats: john.doe@company.com or John Doe <j.doe@company.com>.

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.



### i Note

Connections to an SMTP server over SSL/TLS can cause SSL 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 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.

### i 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).

- Select **Observation Configuration** from the top-right corner of the window.

- 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.

### i 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.

The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a sidebar with navigation links like 'Connector', 'Security Status', 'Alerting', etc. The main area is titled 'Alerting' and shows a sub-section 'Observation Configuration'. This configuration dialog has several sections: 'Observe' (checkboxes for High Availability, Tunnel Health, Service Channels Health, and CPU), 'CPU Threshold Load' (set to 90%), 'CPU Threshold Exceeded' (set to 60 sec), 'Disk' (checkbox checked, threshold set to 1000 MB), 'Certificates' (checkbox checked, threshold set to 30 days), and 'Health Check Interval' (set to 30 sec). At the bottom of the dialog are 'Save' and 'Cancel' buttons. The 'Observation Configuration' tab is highlighted with a red box.

## 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.

## 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](#).

### i 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`.

## 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.

### i Note

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 different categories:

- **Account**: A subaccount configuration was changed.
- **RecoveryAccount**: A disaster recovery 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.
- **SccCustomRoles:** 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 the selected audit entries.

- In the **Audit Type** field, you can select whether to view the audit entries for the following:
  - Any entries
  - Only denied requests
  - Only allowed requests
  - Service started
  - Service stopped
  - Cloud Connector changes
  - High Availability
  - Principal Propagation
- In the **Pattern** field, you can specify a certain string that the detail text of each selected audit entry must contain. The detail text contains information about the user name, requested resource/URL, and the virtual `<host>:<port>`. Wildcards are currently not supported. 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 for which you want to display the audit events.

These filter criteria are combined with a logical AND so that all audit entries that match these criteria are shown. If you have modified one of the criteria, select **Refresh** to display the updated selection of audit events that match the new criteria.

### **i Note**

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.

The  **Check** button checks all files that are filtered by the specified date range.

## Verify the Integrity of Audit Logs

To check the integrity of the audit logs, go to <scc\_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 <scc\_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.

## 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. On the left, there's a navigation sidebar with sections like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', 'Configuration', and 'Audit' (which is currently selected). Below that is another sidebar for the 'trial' subaccount with options like 'Cloud To On-Premise', 'On-Premise To Cloud', 'Monitor', 'Audits' (selected), and 'Log And Trace Files'. The main content area has a 'Subaccount: trial' input field and two tabs: 'Audits' and 'Settings'. Under 'Audits', it shows 'Subaccount Audit Level: Security', 'Cross-Subaccount Audit Level: Security', and 'Automatic Cleanup: Never'. Below this is a detailed audit log table with columns 'Timestamp' and 'Audit Log'.

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:///.....

## 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`.

### **i 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.
  - a. For Microsoft Windows OS: `changeAuditLogPath.bat <desiredLocation>`.
  - b. 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.

## Troubleshooting

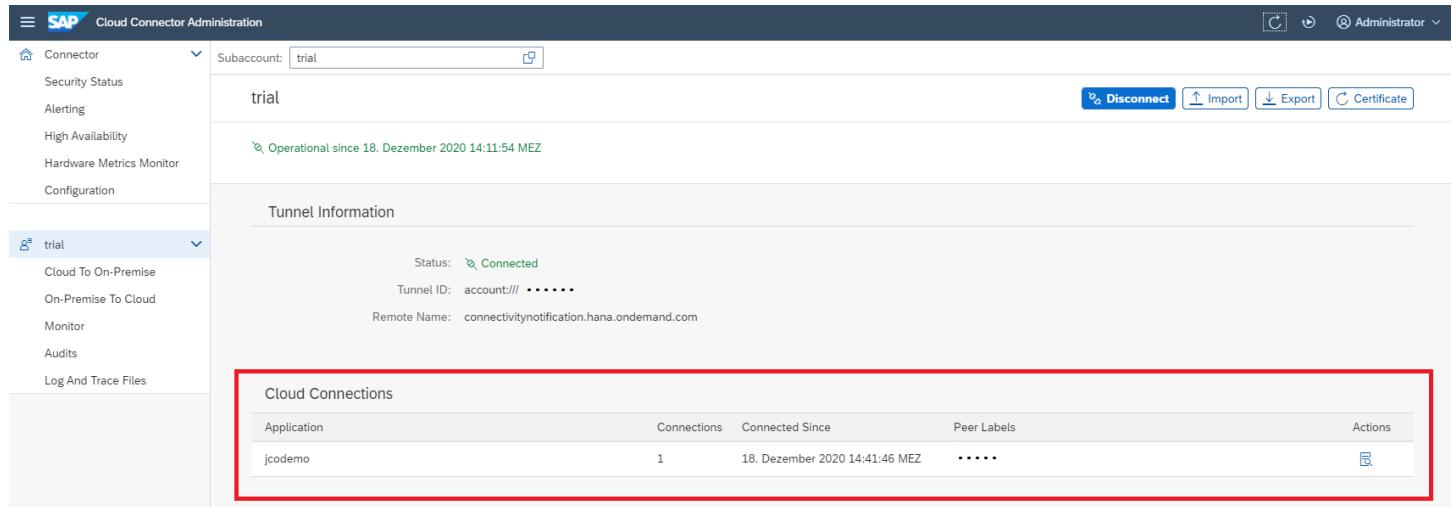
To troubleshoot connection problems, monitor the state of your open tunnel connections in the Cloud Connector, and view different types of logs and traces.

## i Note

For information about a specific problem or an error you have encountered, see [Connectivity Support](#).

## Monitoring

To view a list of all currently connected applications, choose your **Subaccount** from the left menu and go to section **Cloud Connections**:



The screenshot shows the SAP Cloud Connector Administration interface. On the left, there's a navigation sidebar with sections like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', 'Configuration', and 'Log And Trace Files'. Under 'Log And Trace Files', 'trial' is selected. The main area shows a subaccount named 'trial' which has been operational since December 18, 2020, at 14:11:54 MEZ. Below this, 'Tunnel Information' is displayed with status 'Connected', Tunnel ID 'account:/// .....', and Remote Name 'connectivitynotification.hana.ondemand.com'. At the bottom, a table titled 'Cloud Connections' lists one application, 'jcodemo', with 1 connection, established on December 18, 2020, at 14:41:46 MEZ, and peer labels '.....'. The entire 'Cloud Connections' table is highlighted with a red border.

Cloud Connections			
Application	Connections	Connected Since	Peer Labels
jcodemo	1	18. Dezember 2020 14:41:46 MEZ	.....

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*.

**i Note**

Use payload and CPIC tracing at Level 3 carefully and only when requested to do so for support reasons. The trace 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. As of version 2.2, the Cloud Connector supports an 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. See [Secure the Activation of Traffic Traces](#).

- SSL Trace:** When the SSL trace is activated, the `ljs_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:	<input type="text" value="Information"/>
Other Loggers:	<input type="text" value="Information"/>
CPIC Trace Level:	<input type="text" value="0"/>
Payload Trace:	<input type="checkbox"/>
SSL Trace:	<input type="checkbox"/>
Automatic Cleanup:	<input type="text" value="After 365 Days"/>

**Save**    [Cancel](#)

## Change the Location of Trace Files

As of Cloud Connector 2.14 you can move trace files to a different location.

**i Note**

JVM-related files will remain in the standard location `log`.

**i 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.

a. For Microsoft Windows OS: `changeLogAndTracePath.bat <desiredLocation>`.

b. 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.

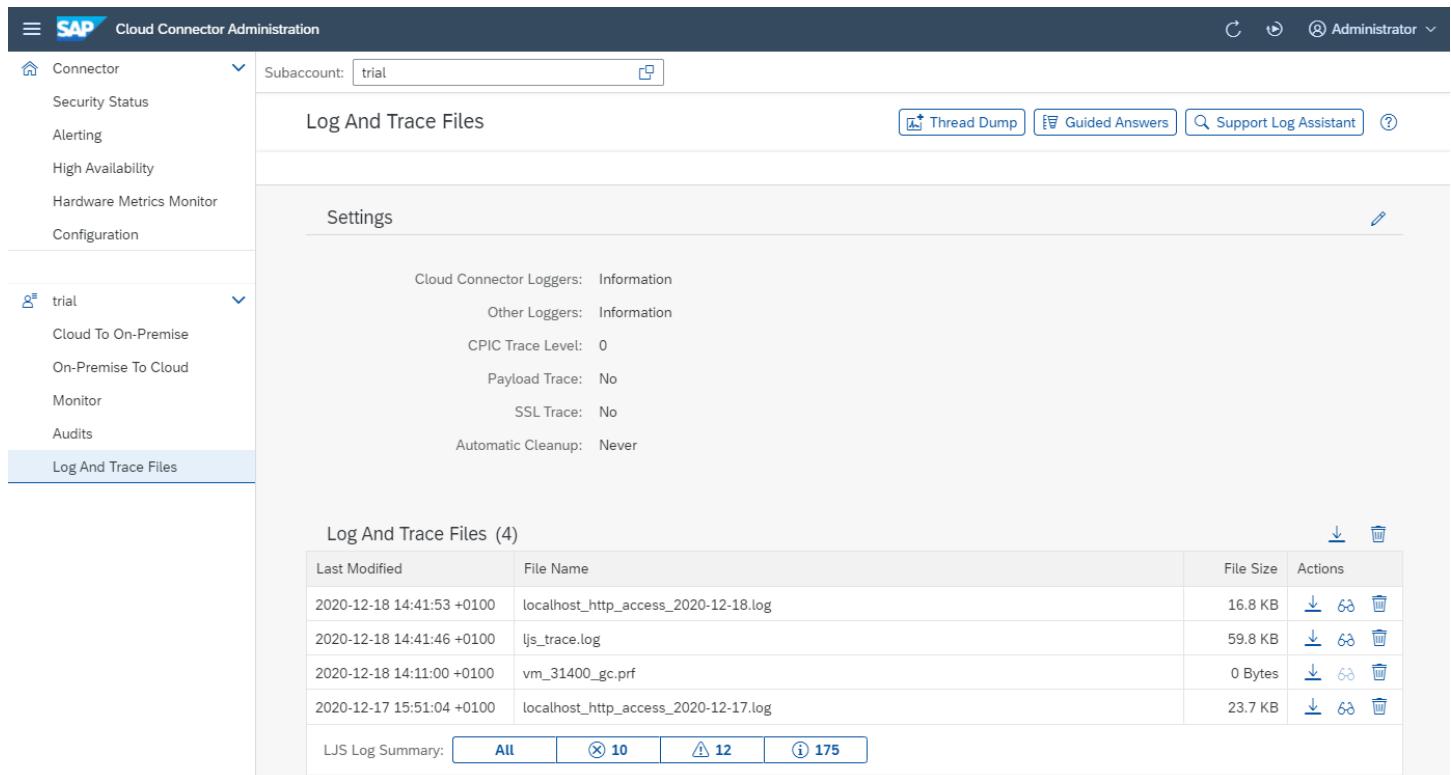
4. Start the Cloud Connector again.

### 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 shows the SAP Cloud Connector Administration interface. The top navigation bar includes the SAP logo, 'Cloud Connector Administration', and a user 'Administrator'. The left sidebar has a tree view with 'Connector' expanded, showing 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. Under 'Connector', 'trial' is selected, showing 'Cloud To On-Premise', 'On-Premise To Cloud', 'Monitor', and 'Audits'. The 'Log And Trace Files' tab is selected. The main content area has a 'Log And Trace Files' header with buttons for 'Thread Dump', 'Guided Answers', and 'Support Log Assistant'. Below this is a 'Settings' section with fields: 'Cloud Connector Loggers: Information', 'Other Loggers: Information', 'CPIC Trace Level: 0', 'Payload Trace: No', 'SSL Trace: No', and 'Automatic Cleanup: Never'. At the bottom is a table titled 'Log And Trace Files (4)' with columns 'Last Modified', 'File Name', 'File Size', and 'Actions'. The table lists four files: 'localhost\_http\_access\_2020-12-18.log' (16.8 KB), 'ljs\_trace.log' (59.8 KB), 'vm\_31400\_gc.prf' (0 Bytes), and 'localhost\_http\_access\_2020-12-17.log' (23.7 KB). Each row has download and delete icons. Below the table is a 'LJS Log Summary' with buttons for 'All', 'X 10', '▲ 12', and 'i 175'.

Last Modified	File Name	File Size	Actions
2020-12-18 14:41:53 +0100	localhost_http_access_2020-12-18.log	16.8 KB	
2020-12-18 14:41:46 +0100	ljs_trace.log	59.8 KB	
2020-12-18 14:11:00 +0100	vm_31400_gc.prf	0 Bytes	
2020-12-17 15:51:04 +0100	localhost_http_access_2020-12-17.log	23.7 KB	

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.

**i 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.

**i 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.
- <*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

# 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](#)

[Document Administrator Roles](#)

[Document Communication Channels](#)

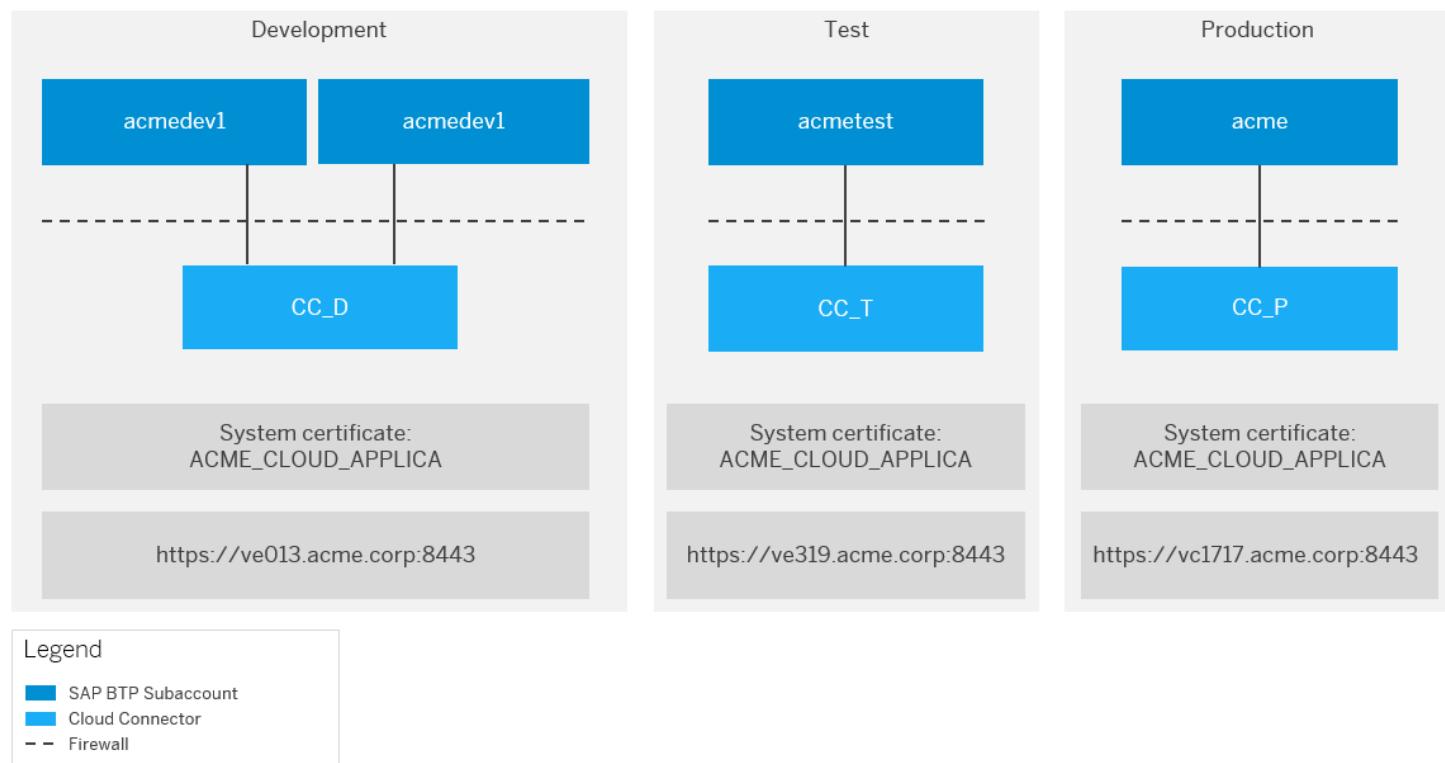
[Define Project and Development Guidelines](#)

[Define How to Set a Cloud Application Live](#)

## 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](#)

## Document Administrator Roles

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

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		
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](#)

## 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](#)

## 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](#)

## 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](#)

## Configuring Backup

Find an overview of backup procedures for the Cloud Connector.

<a href="#">Configuration Backup</a>	Backup and restore your Cloud Connector configuration via the administration UI.
<a href="#">Backup</a>	Manage the Cloud Connector's configuration backup via REST API.
<a href="#">Backup And Restore Configuration</a>	Example: Backup and restore the Cloud Connector configuration via REST API.

## 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
<a href="#">Application Layer</a>	<ul style="list-style-type: none"> <li>Frontend security</li> <li>Security standard-based application development</li> </ul>
<a href="#">Service Layer</a>	<ul style="list-style-type: none"> <li>Identity and access management</li> <li>Data protection</li> <li>Regulatory compliance management</li> </ul>
<a href="#">Cloud Infrastructure Layer</a>	<ul style="list-style-type: none"> <li>Network infrastructure and communication</li> <li>Sandboxing</li> <li>Intrusion detection and prevention</li> </ul>
<a href="#">Physical and Environmental Layer</a>	<ul style="list-style-type: none"> <li>Strict physical access control</li> <li>High availability and disaster recovery capabilities</li> </ul>

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](#)

## 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.

## Service Layer

You can use SAP BTP Connectivity to securely integrate cloud applications with systems running in isolated customer networks.

### Overview

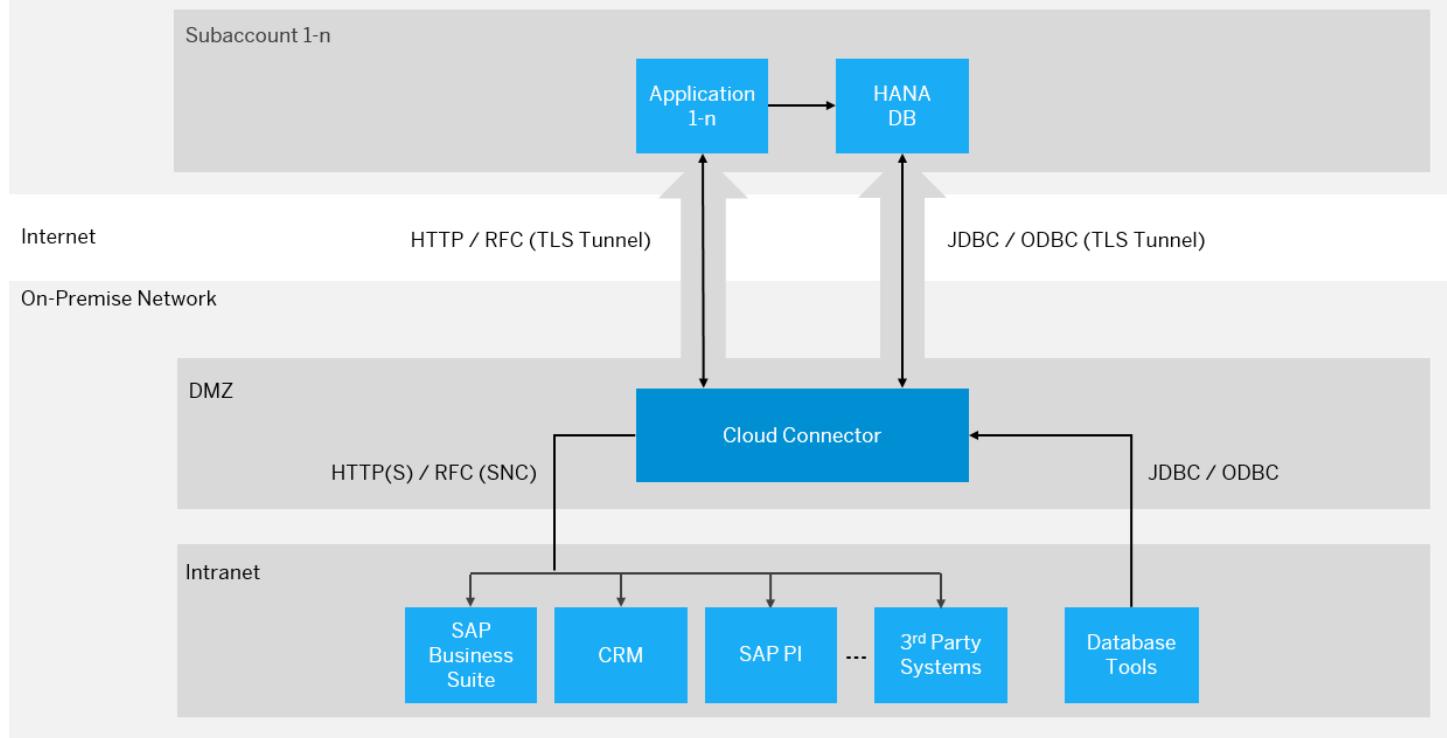
After installing the Cloud Connector as integration agent in your on-premise network, you can use it to establish a persistent TLS tunnel to SAP BTP subaccounts.

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](#).
- 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](#)

[Inbound Connectivity](#)

[Outbound Connectivity](#)

[Audit Log](#)

## 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](#)).
- 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](#)

# 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](#)

## 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](#)

## 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](#)).

## 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](#)

## 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.

## 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

## 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.

Network Zone

Administration UI

High Availability

On-Premise Configuration

OS-Level Protection

Protocol Security

Audit Logging

Instances

Please note that image maps are not interactive in PDF output.

## Recommended Actions

Topic	Description	Recommended Action
<a href="#">Network Zone</a> <a href="#">Back to Topics</a>	<p>Depending on the needs of the project, the Cloud Connector can be either set up in the <b>DMZ</b> and operated centrally by the IT department or set up in the <b>intranet</b> and operated by the line-of-business.</p>	<p>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.</p> <p>Set up trust between the on-premise system and the Cloud Connector, and only accept requests from trusted Cloud Connectors in the system.</p>
<a href="#">OS-Level Protection</a> <a href="#">Back to Topics</a>	<p>The Cloud Connector is a security-critical component that handles the inbound access from SAP BTP applications to systems of an on-premise network.</p> <p><b>Methods to secure the operating system</b>, on which the Cloud Connector is running, should be applied.</p>	<p>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.</p> <p>Use the machine which runs the Cloud Connector only for this purpose and don't reuse it for other scenarios.</p> <p>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.</p> <p>Turn on the audit log on operating system level to monitor the file operations.</p>
<a href="#">Administration UI</a> <a href="#">Back to Topics</a>	<p>After installation, the Cloud Connector provides an initial user name and password and forces the user (<b>Administrator</b>) to <b>change the password</b> upon initial logon.</p>	<p>Change the password of the <b>Administrator</b> user immediately after installation. Choose a strong password for the user (see also <a href="#">Recommendations for Secure Setup</a>).</p> <p>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 <a href="#">Use LDAP for Authentication</a>).</p>

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 <b>X.509 certificate</b> as SSL 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 ).</p> <p>For high-security scenarios, limit the access to the Cloud Connector administration UI to localhost (see also <a href="#">Recommendations for Secure Setup</a>).</p>
<a href="#">Audit Logging</a>  Back to <a href="#">Topics</a>	<p>For end-to-end traceability of configuration changes in the Cloud Connector, as well as communication delivered by the Cloud Connector, <b>switch on audit logging</b> for productive scenarios.</p>	<p>Switch on audit logging in the Cloud Connector: set audit level to "All" (see <a href="#">Recommendations for Secure Setup</a> and <a href="#">Manage Audit Logs</a>)</p> <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> <p>To gain end-to-end traceability, you should switch on audit logging also in the connected on-premise systems.</p>
<a href="#">High Availability</a>  Back to <a href="#">Topics</a>	<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 <b>second (redundant) Cloud Connector</b> in place.</p>	<p>Use the high availability feature of the Cloud Connector for productive scenarios (see <a href="#">Install a Failover Instance for High Availability</a>).</p>
<a href="#">Supported Protocols</a>  Back to <a href="#">Topics</a>	<p><b>HTTP, HTTPS, RFC and RFC over SNC</b> 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 <a href="#">Set Up Trust for Principal Propagation</a>).</p>
<a href="#">Configuration of On-Premise Systems</a>  Back to <a href="#">Topics</a>	<p>When configuring the access to an internal system in the Cloud Connector, map physical host names to virtual host names to <b>prevent exposure of information</b> on physical systems to the cloud.</p>	<p>Use hostname mapping of exposed on-premise systems in the access control of the Cloud Connector (see <a href="#">Configure Access Control (HTTP)</a> and <a href="#">Configure Access Control (RFC)</a>).</p>

Topic	Description	Recommended Action
	When configuring the access to an internal system, <b>restrict access</b> to those resources which are actually required by the cloud applications. Do not expose the complete system.	Narrow access to on-premise systems to resources required by the relevant cloud applications in the access control of the Cloud Connector (see <a href="#">Configure Access Control (HTTP)</a> and <a href="#">Configure Access Control (RFC)</a> ).
	To allow <b>access only for trusted applications</b> of your SAP BTP subaccount to on-premise systems, configure the list of trusted applications in the Cloud Connector.	Narrow the list of cloud applications which are allowed to use the on-premise tunnel to the ones that need on-premise connectivity (see <a href="#">Set Up Trust for Principal Propagation</a> ).
<a href="#">Cloud Connector Instances</a> <a href="#">Back to Topics</a>	<p>You can connect a single Cloud Connector instance to multiple SAP BTP subaccounts. Subaccounts can be created by SAP BTP users as a self service. Different subaccounts are often used to <b>separate development, test and production</b>.</p> <p>Do not mix productive Cloud Connector usages with development or test scenarios.</p>	Use different Cloud Connector instances to separate productive and non-productive scenarios.

## Related Information

[Recommendations for Secure Setup](#)

[Secure the Activation of Traffic Traces](#)

## 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](#). Before upgrading, please check the [Prerequisites](#) and make sure your environment fits the new version. We recommend that you create a [Configuration Backup](#) 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](#).

## Microsoft Windows OS

1. Uninstall the Cloud Connector as described in [Uninstallation](#) 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](#).
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.

## 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 SSL certificates contained in the JVM, 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.

**i 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.

**i 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:

- o in csh/tcsh:

```
setenv JAVA_HOME /opt/sap/sapjvm_8
```

- o 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

### i Note

If you use your own CA certificates for the Email configuration (see [Alerting](#)) or for LDAP (see [Use LDAP for Authentication](#)), you must reimport them to the JVM trust store as described there.

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.

## 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](#)

# 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](#) 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 JRE versions are supported to run the Cloud Connector?

		Supported JRE Version			
		6	7	8	11
Cloud Connector Version	< 2.7.2	Yes	Yes	No	No
	= 2.7.2	Yes	Yes	Yes	No
	>= 2.8	No	Yes	Yes	No
	>= 2.12.3	No	No	Yes	No
	>= 2.14.0	No	No	Yes	Yes

## ! Restriction

Support for Java 7 has been discontinued. For more information, see [Prerequisites](#).

## → Tip

We recommend that you always use the latest supported JRE 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](#). For more details on which network zones to choose for the Cloud Connector setup, see [Network Zones](#).

**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](#).

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

### **i 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](#).

**Can I send push messages from an on-premise system to the SAP BTP through the Cloud Connector?**

No, this is not supported by the Cloud Connector.

**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](#).

**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](#).

**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-to-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](#).

### 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 Authentication](#).

### 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](#)) or via REST API (see [Backup](#)).

### 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](#).

## For how long does SAP continue to support older Cloud Connector versions?

Each Cloud Connector version is supported for 12 months, which means the cloud side infrastructure is guaranteed to stay compatible with those versions.

After that time frame, compatibility is no longer guaranteed and interoperability could be dropped. Furthermore, after an additional 3 month, the next feature release published after that period will no longer support an upgrade from the deprecated version as a starting release.

## 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”:

The screenshot shows the SAP Cloud Connector interface. The left sidebar has sections for Overview, Applications (Java Applications, HTML5 Applications, HANA XS Applications), Subscriptions, Services, Persistence, and Connectivity. The main area is titled 'System Status' and contains two boxes: 'JAVA' and 'HTML5'. The JAVA box shows 'Overall Health' with a yellow diamond icon and 'N/A' status. It also shows '15 Applications' and '15 Stopped'. The HTML5 box shows 'Overall Heal' with a green checkmark icon and 'OK' status.

## 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](#).

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.

#### **i** 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

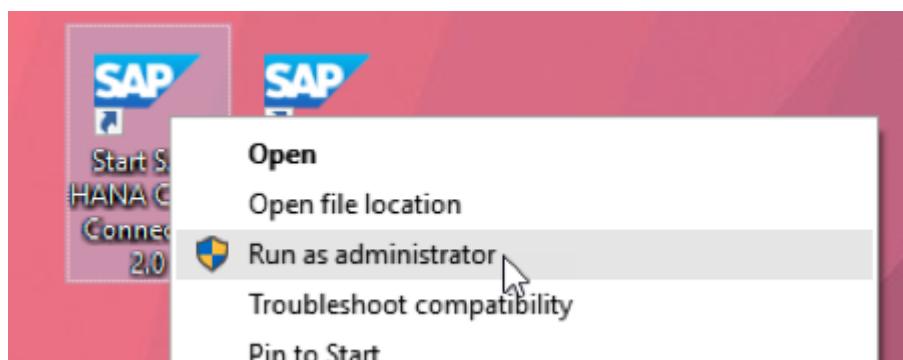
### How do I fix the "Could not open Service Manager" error message?

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.

#### **i 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/>.

# 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](#)

[Security and Session](#)

[User Roles](#)

[Return Codes](#)

[Hypertext Application Language](#)

[Available REST APIs](#)

[Sample Code](#)

## 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. 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, F
```

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.

### i 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](#)

## 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.

### i 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](#)

## User Roles

As of Cloud Connector 2.12, 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](#)

## 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.

### i Note

The error texts depend on the request locale.

Back to [Top](#)

## 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](#)

## Available REST APIs

The Cloud Connector provides APIs for:

[Monitoring tasks](#)

[Configuration tasks](#)

Back to [Top](#)

## 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](#)