



PUBLIC
2024-01-25

SAP BTP Connectivity for the Neo Environment

Content

1	Connectivity for the Neo Environment.	3
1.1	What Is Connectivity for the Neo Environment?.	8
1.2	What's New for Connectivity.	11
	2021 Connectivity (Archive).	11
	2020 Connectivity (Archive).	29
	2019 Connectivity (Archive).	38
	2018 Connectivity (Archive).	45
	2017 Connectivity (Archive).	49
1.3	Administration.	53
	Managing Destinations.	54
	HTTP Destinations.	91
	RFC Destinations.	114
	LDAP Destinations.	126
	Mail Destinations.	128
	Principal Propagation.	130
	Multitenancy in the Connectivity Service.	132
	Configuring Backup.	135
1.4	Development.	135
	Consuming the Connectivity Service (Java).	135
	Consuming the Connectivity Service (HANA XS).	217
	Consuming the Destination Configuration Service	231
1.5	Cloud Connector (Neo Environment).	232
	Installation.	238
	Configuration.	277
	Administration.	516
	Security.	589
	Upgrade	600
	Update the Java VM	602
	Uninstallation.	603
	Frequently Asked Questions	604
1.6	Connectivity via Reverse Proxy.	614
1.7	Connectivity Support.	615
	Release and Maintenance Strategy.	617

1 Connectivity for the Neo Environment

SAP BTP Connectivity: overview, features, restrictions.

ⓘ Note

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

Content

In this Topic

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

Overview

Features

Restrictions

- [Overview](#) [page 4]
- [Features](#) [page 5]
- [Restrictions](#) [page 5]

In this Guide

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

Neo Environment

Cloud Connector

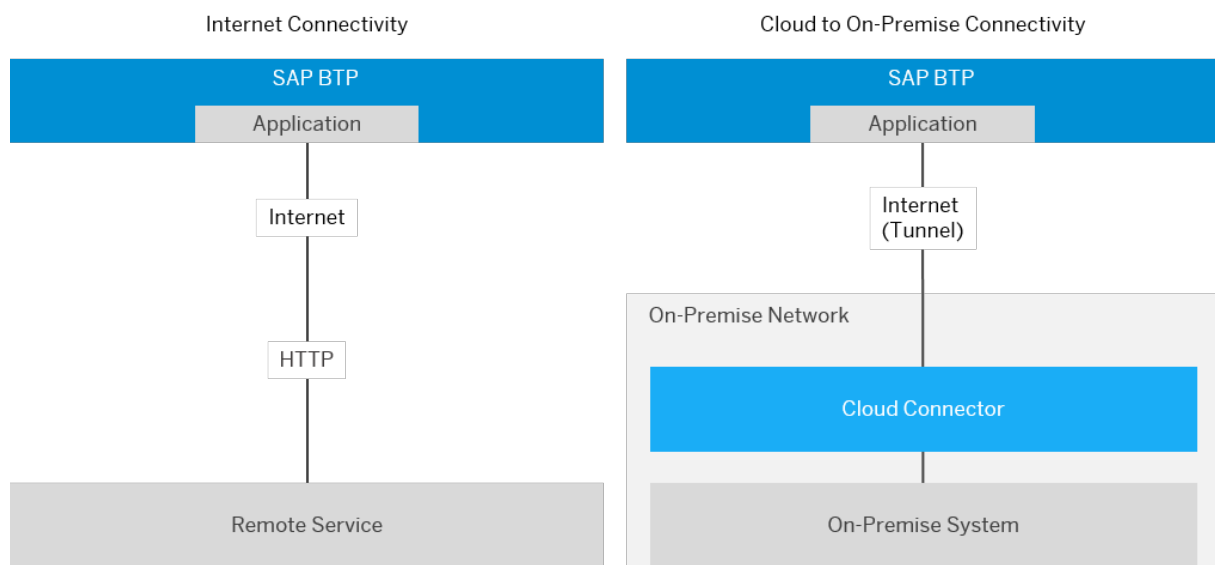
Support

- [What Is Connectivity for the Neo Environment?](#) [page 8]
- [Cloud Connector \(Neo Environment\)](#) [page 232]
- [Connectivity Support](#) [page 615]

Overview

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

- 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.



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

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

For inbound connections (calling an application or service on SAP BTP from an external source), you can use Cloud Connector [service channels \[page 491\]](#) (on-premise connections) or the respective [host](#) of your SAP BTP region (Internet connections).

Back to [Content \[page 3\]](#)

Features

The Connectivity service supports the following protocols and scenarios:

Protocol	Scenario
HTTP(S)	<p>Exchange data between your cloud application and Internet services or on-premise systems.</p> <ul style="list-style-type: none">• Create and configure HTTP destinations to make Web connections.• Connect to on-premise systems via HTTP, using the Cloud Connector.
RFC	<p>Invoke on-premise ABAP function modules via RFC.</p> <ul style="list-style-type: none">• Create and configure RFC destinations.• Make connections to back-end systems via RFC, using the Cloud Connector.
TCP	<p>Access on-premise systems via TCP-based protocols using a SOCKS5 proxy.</p>
LDAP	<p>Enables LDAP-based user management if you are operating an LDAP server within your network.</p>
Mail Protocols	<p>Applications use the standard <code>javax.mail</code> API. The e-mail provider and e-mail account are configured using mail destinations.</p> <ul style="list-style-type: none">• The SMTP protocol allows you to send e-mail messages from your Web applications using e-mail providers that are accessible on the Internet, such as Google Mail (Gmail).• The IMAP and POP3 protocols let you retrieve e-mails from the mailbox of your e-mail account.

Back to [Content \[page 3\]](#)

Restrictions

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

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

[Neo Environment \[page 6\]](#)

[Cloud Connector \[page 7\]](#)

Note

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

General

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

Back to [Restrictions \[page 5\]](#)

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.
TCP	You can use TCP-based communication for any client that supports SOCKS5 proxies.
LDAP	Enables LDAP-based user management if you are operating an LDAP server within your network.

Back to [Restrictions \[page 5\]](#)

Neo Environment

Topic	Restriction
E-Mail	<ul style="list-style-type: none"> You cannot communicate with an e-mail provider via an unencrypted SMTP protocol on port 25. Fetches e-mail is not scanned for viruses. Sending e-mail with attachments using <code>javax.activation.DataHandler</code> works with SAP BTP SDK for Java EE 6 Web Profile. Mail destinations can be configured only on application level. That is, configuration on a subscription or customer subaccount level is not supported. For SAP BTP SDK for Java Web and SAP BTP SDK for Java EE 6 Web Profile: Applications must use the <code>javax.mail</code> version that is provisioned by the SAP BTP runtime (see Connectivity and Destination APIs [page 136]). Applications must not include the <code>javax.mail</code> library as part of the web archive.

Back to [Restrictions \[page 5\]](#)

Cloud Connector

Topic	Restriction
Scenarios	To learn in which system landscapes you can set up the Cloud Connector, see Extended Scenarios [page 236] .
Installation	To check all software and hardware restrictions for working with the Cloud Connector, see Prerequisites [page 239] .

Back to [Restrictions \[page 5\]](#)

Back to [Content \[page 3\]](#)

Related Information

[What's New for Connectivity \[page 11\]](#)

[Connectivity via Reverse Proxy \[page 614\]](#)

1.1 What Is Connectivity for the Neo Environment?

Use SAP Connectivity service for your application in the Neo environment. Learn about destination management, connectivity scenarios, and required user roles.

Note

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

Content

In this Topic

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

Destinations

Scenarios

User Roles

- [Destinations \[page 8\]](#)
- [User Roles \[page 9\]](#)
- [Scenarios \[page 9\]](#)

In this Guide

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

Administration

Development

- [Administration \[page 53\]](#)
- [Development \[page 135\]](#)

Destinations

To use of the Connectivity service, you must first create and configure destinations, using the corresponding communication protocol and other destination properties.

You have several options to create and edit destinations, see [Managing Destinations \[page 54\]](#).

To learn how to configure a destination for a specific protocol, see:

- [HTTP Destinations \[page 91\]](#)
- [RFC Destinations \[page 114\]](#)
- [LDAP Destinations \[page 126\]](#)
- [Mail Destinations \[page 128\]](#)

Back to [Content \[page 8\]](#)

Scenarios

Scenario	More Information
Connect Web applications and external servers via HTTP	Consume Internet Services (Java Web or Java EE 6 Web Profile) [page 156]
Make connections between Web applications and on-premise backend services via HTTP	Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 171]
Connect Web applications and on-premise backend services via RFC	Invoke ABAP Function Modules in On-Premise ABAP Systems [page 195]
Use LDAP-based user authentication for your cloud application	LDAP Destinations [page 126]
Access on-premise systems via TCP-based protocols using a SOCKS5 proxy	Using the TCP Protocol for Cloud Applications [page 205]
Send and fetch e-mail via mail protocols	Sending and Fetching E-Mail [page 208]

Back to [Content \[page 8\]](#)

User Roles

The following user groups are involved in the end-to-end use of the Connectivity service:

- **Application operators** - are responsible for productive deployment and operation of an application on SAP BTP. They are also responsible for configuring destinations and certificates for the remote connections that an application may need, see [Administration \[page 53\]](#).
- **Application developers** - create a connectivity-enabled SAP BTP application by using the Connectivity service API, see [Development \[page 135\]](#).
- **IT administrators** - set up the connectivity to SAP BTP in your on-premise network, using the [Cloud Connector \[page 232\]](#).

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

Task Types



Operator



Developer



Operator and/or Developer

To perform connectivity tasks in the Neo environment, the following user roles and authorizations apply:

Connectivity UI (Level) in the Cockpit	Subaccount Roles	OAuth Scopes
Cloud Connectors (manage)	Administrator Cloud Connector Admin	manageSCCTunnels
Cloud Connectors (view)	A role containing the permission <code>readSCCTunnels</code> , for example, the predefined role Cloud Connector Admin.	readSCCTunnels
Destinations (<i>subaccount</i> level)	Administrator	manageSCCTunnels
Destinations (<i>application</i> level)	Administrator Developer	manageSCCTunnels
Destinations (<i>subscription</i> level)	Administrator	manageSCCTunnels
Destinations (<i>service</i>)	Administrator	manageSCCTunnels

For more information on the configuration levels available for destination management, see [Managing Destinations \[page 54\]](#) (section *Configuration Levels (HTTP and RFC)*).

See also:

[Managing Member Authorizations in the Neo Environment](#)

[Managing Roles](#)

[Platform Scopes](#)

Back to [Content \[page 8\]](#)

Related Information

[Prerequisites and Restrictions](#)

[Administration \[page 53\]](#)

[Development \[page 135\]](#)

[Security](#)
[Multitarget Applications for the Neo Environment](#)
[Prerequisites and Restrictions](#)

1.2 What's New for Connectivity

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

[What's New for Connectivity](#)

Related Information

[2021 Connectivity \(Archive\) \[page 11\]](#)
[2020 Connectivity \(Archive\) \[page 29\]](#)
[2019 Connectivity \(Archive\) \[page 38\]](#)
[2018 Connectivity \(Archive\) \[page 45\]](#)
[2017 Connectivity \(Archive\) \[page 49\]](#)


1.2.1 2021 Connectivity (Archive)

2021



Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Cloud Connector - SAP Cloud PKI	<p>SAP Cloud PKI (public key infrastructure) is enabled for technical communication between Cloud Connector and Connectivity service.</p> <ul style="list-style-type: none"> The change is transparent for the Cloud Connector - as soon as you renew your subaccount certificate in the Cloud Connector, the newly issued X.509 client certificate will be part of SAP Cloud PKI. If you are using Connectivity proxy software components as part of your solution, make sure you use version 2.4.1 or higher. For scenarios with termination in the ingress, version 2.3.1 of the Connectivity proxy is sufficient. 	Info only	New	2021-12-02
Connectivity	Integration Suite	Cloud Foundry Neo	Destination Service - Mail Destinations	<p>You can now configure destinations of type MAIL with any OAuth-based authentication type as available for HTTP destinations, including the option for mTLS via X.509 client certificate.</p> <div> <p>⚠ Restriction</p> <p>The Mail Java API (Neo environment) does not provide the <code>javax.mail.Session</code> object out of the box. It must be configured manually.</p> </div>	Info only	New	2021-12-02


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

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

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

1.2.2 2020 Connectivity (Archive)


2020

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

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

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

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

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

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

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

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

1.2.3 2019 Connectivity (Archive)

2019

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

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


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

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

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

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

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

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

1.2.4 2018 Connectivity (Archive)

2018

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

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

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo	JCo Runtime for SAP Cloud Platform - Features	Support of the RFC fast serialization. Depending on the exchanged parameter and data types, the performance improvements for RFC communication can reach multiple factors. See SAP note 2372888 (prerequisites) and Parameters Influencing Communication Behavior [page 123] (JCo configuration in SAP Cloud Platform).	Changed	2018-12-06
Connectivity	Integration	Neo	JCo Runtime for SAP Cloud Platform - Information	Local runtimes on Windows must install the VS 2013 redistributables for x64, instead of VS 2010.	Changed	2018-12-06
Connectivity	Integration	Neo Cloud Foundry	Cloud Connector Fixes	Release of Cloud Connector 2.11.3: <ul style="list-style-type: none"> An issue in RFC communication could cause the trace entry <i>com.sap.scc.jni.CpicCommunicationException: no SAP ErrInfo available</i> when the network is slow. This issue has been fixed. The Windows service no longer runs in <i>error 1067</i> when stopped by an administrator. In previous releases, the connection between a shadow and a master instance occasionally failed at startup and produced an empty error message. This issue has been fixed. The Cloud Connector does not cache Kerberos tokens in the protocol handler any more, as they are one-time tokens and cannot be reused. For HTTP access control entries, you can configure resources containing a # character. 	Changed	2018-12-06

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo Cloud Foundry	Cloud Connector Enhancements	<p>Release of Cloud Connector 2.11.3:</p> <ul style="list-style-type: none"> If the user <i>sapadm</i> exists on a system, the installation on Linux assigns it to the <i>sccgroup</i>, which is a prerequisite for solution management integration to work properly, see Configure Solution Management Integration [page 506]. Restoring a backup has been improved. See Configuration Backup [page 511]. The HTTP session store size has been reduced. You can handle higher loads with a given heap size. Cipher suite configuration has been improved. Also, there is a new security status entry for cipher suites, see Recommendations for Secure Setup [page 272]. 	Changed	2018-12-06
Connectivity	Integration	Neo	HTTP Destinations	<p>The <i>OAuth2 Client Credentials</i> grant type is supported by the Destinations editor in the SAP Cloud Platform cockpit as well as by the client Java APIs <code>ConnectivityConfiguration</code>, <code>AuthenticationHeaderProvider</code> and <code>HttpDestination</code>, available in SAP Cloud Platform Neo runtimes.</p> <p>See OAuth Client Credentials Authentication.</p>	Changed	2018-10-11
Connectivity	Integration	Cloud Foundry	User Propagation	<p>The connectivity service supports the SaaS application subscription flow and can be declared as a dependency in the get dependencies subscription callback, also via MTA (multi-target)-bundled applications.</p> <p>See Consuming the Connectivity Service (Cloud Foundry Environment) and Configure Principal Propagation via User Exchange Token (Cloud Foundry Environment).</p>	Changed	2018-09-27
Connectivity	Integration	Neo Cloud Foundry	Cloud Connector 2.11.2	<p>Release of Cloud Connector 2.11.2</p> <ul style="list-style-type: none"> SNC configuration now provides the value of the environment variable <code>SECUDIR</code>, which you need for the usage of the SAP Cryptographic Library (SAPCRYPTOLIB). See Initial Configuration (RFC). On Linux, the RPM (Red Hat Package Manager) now ensures that the configuration of the interaction with the SAP Host Agent (used for the Solution Manager integration) is adjusted. See Configure Solution Management Integration. The Cloud Connector shadow instance now provides a configuration option for the connection and request timeout that may occur during health check against the master instance. See Master and Shadow Administration. 	Changed	2018-08-16

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

1.2.5 2017 Connectivity (Archive)

Archived release notes for 2017 and older.

28 September 2017 - Connectivity

New

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

3 August 2017 - Connectivity

Enhancement

Cloud Connector

Release of SAP Cloud Platform Cloud Connector 2.10.1.

- The URLs of HTTP requests can now be longer than 4096 bytes.
 - SAP Solution Manager can be integrated with one click of a button if the host agent is installed on a Cloud Connector machine. See the *Solution Management* section in [Monitoring \[page 546\]](#).
 - The limitation that only 100 subaccounts could be managed with the administration UI has been removed. See [Managing Subaccounts \[page 291\]](#).
-

Fix

Cloud Connector

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

25 May 2017 - Connectivity

Enhancement

Cloud Connector

Release of SAP HANA Cloud connector 2.10.0.1

- Support of connectivity to an SAP Cloud Platform Cloud Foundry environment.
- Support of direct connectivity with S/4HANA Cloud systems. You can open a Service Channel to an S/4HANA Cloud system in order to use Communication Scenarios requiring RFC communication to S/4HANA Cloud. See [Configure a Service Channel for RFC \[page 498\]](#).
- Support of arbitrary protocols via the possibility to configure a TCP access control entry. SAP Cloud Platform Connectivity is offering a SOCKS5 proxy, with which you can address such exposed hosts. See [Using the TCP Protocol for Cloud Applications \[page 205\]](#).
- Support for disaster recovery events of SAP Cloud Platform regions. For each subaccount you can configure a disaster recovery subaccount for a disaster region. In case of a disaster, the disaster recovery account can be switched active immediately using the exact same configuration. See [Configure a Disaster Recovery Subaccount \[page 299\]](#).
- In the access control settings you can add further constraints for RFC based communication to ABAP systems: an administrator can configure, which clients shall be exposed and can define which users should not be able to access the system via Cloud Connector. See [Configure Access Control \(RFC\) \[page 350\]](#).
- You can generate self-signed certificates for CA and system certificate so that you can setup demo scenarios with principal propagation without the need of using lengthy openssl or keytool command sequences. See [Configure a CA Certificate \[page 308\]](#) and [Initial Configuration \(HTTP\) \[page 285\]](#).
- A first set of monitoring HTTP APIs have been introduced: The state of all subaccount connections, a back-end connection monitor and a performance overview monitor. See [Monitoring APIs \[page 556\]](#).
- On Windows platforms, Cloud Connector 2.10 now requires Visual Studio 2013 runtime libraries.

Fix

Cloud Connector

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

11 May 2017 - Connectivity

New

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

30 March 2017 - Connectivity

Enhancement

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

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

Fix

- When configuring authentication for LDAP, the alternate host settings are no longer ignored.
 - The email configuration for alerts is processing correctly the user and password for access to the email server.
 - Some servers used to fail to process HTTP requests when using the HTTP proxy approach ([HTTP Proxy for On-Premise Connectivity \[page 153\]](#)) on the SAP Cloud Platform side.
 - A bottleneck was removed that could lengthen the processing times of requests to exposed back-end systems under high load when using principal propagation.
 - The Cloud Connector accepts passwords that contain the '\$' character when using authentication-mode password.
-

16 March 2017 - Connectivity

Enhancement

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

Fix

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

Older Release Notes

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

1.3 Administration

Manage destinations and authentication for applications in the Neo environment.

Task	Description
Managing Destinations [page 54]	Create and configure destinations. You can use destinations for outbound communication between a cloud application and a remote system.
HTTP Destinations [page 91]	Create and configure HTTP destinations.
RFC Destinations [page 114]	Create and configure RFC destinations.
LDAP Destinations [page 126]	Create and configure LDAP destinations.
Mail Destinations [page 128]	Create and configure mail destinations.
Principal Propagation [page 130]	Use principal propagation to forward the identity of cloud users to a back-end system (single sign-on).
Multitenancy in the Connectivity Service [page 132]	Manage destinations for multitenancy-enabled applications that require a connection to a remote service or on-premise application.
Configuring Backup [page 135]	Find an overview of backup procedures for your destination configurations.

1.3.1 Managing Destinations

Overview

Destinations are used for the outbound communication of a cloud application to a remote system and contain the required connection information. They are represented by symbolic names that are used by cloud applications to refer to a remote connection.

The Connectivity service resolves the destination at runtime based on the symbolic name provided. The result is an object that contains customer-specific configuration details, for example, the URL of the remote system or service, the authentication type, and the required credentials.

To configure a destination, you can use files with extension `.props`, `.properties`, `.jks`, and `.txt`, as well as files with no extension.

Destination Names

A destination name must be unique for the current application. It must contain only alphanumeric characters, underscores, and dashes. The maximum length is 200 characters.

Destination Types

The currently supported destination types are **HTTP**, **RFC**, **LDAP** and **Mail**.

- [HTTP Destinations \[page 91\]](#) - provide data communication via the HTTP protocol and are used for both Internet and on-premise connections.
- [RFC Destinations \[page 114\]](#) - make connections to ABAP on-premise systems via RFC protocol using the Java Connector (JCo) as API.
- [LDAP Destinations \[page 126\]](#) - enable LDAP-based user management if you are operating an LDAP server within your network.
- [Mail Destinations \[page 128\]](#) - specify an e-mail provider for sending and retrieving e-mails via SMTP, IMAP, and POP3 protocols.

Configuration Tools

To configure and use a destination to connect your cloud application, you can use one of the following tools:

- [Configure Destinations from the Eclipse IDE \[page 63\]](#)
- [Configure Destinations from the Cockpit \[page 76\]](#)
- [Configure Destinations from the Console Client \[page 55\]](#)

Configuration Level (HTTP and RFC)

Destinations can be simultaneously configured on three levels: application, consumer subaccount, and subscription. This means it is possible to have one and the same destination on more than one configuration level.

- *Application level* - The destination is related to an application and its relevant provider subaccount. It is, though, independent from the consumer subaccount in which the application is running.
- *Consumer subaccount level* - The destination is related to a particular subaccount.
- *Subscription level* - The destination is related to the triad <Application, Provider Subaccount, Consumer Subaccount>.

The runtime tries to resolve a destination in the following order: *Subscription level* → *Consumer subaccount level* → *Provider application level*.

For more information about the usage of consumer subaccount, provider subaccount, and provider application, see [Configure Destinations from the Console Client \[page 55\]](#).

Configuration Cache

- Destination configuration files and Java keystore (JKS) files are cached at runtime. The cache expiration time is set to a small time interval (currently around 4 minutes). This means that once you update an existing destination configuration or a JKS file, the application needs about 4 minutes until the new destination configuration is applied. To avoid this waiting time, the application can be restarted on the cloud; following the restart, the new destination configuration takes effect immediately.
- When you configure a destination for the first time, it takes effect immediately.
- If you change a mail destination, the application needs to be restarted before the new configuration becomes effective.

Examples

You can find examples in the SDK package that you previously downloaded from <http://tools.hana.ondemand.com>.

Open the SDK location and go to `/tools/samples/connectivity`. This folder contains a standard `template.properties` file, `weather` destination, and `weather.destinations.properties` file, which provides all the necessary properties for uploading the `weather` destination.

1.3.1.1 Configure Destinations from the Console Client

As an application operator, you can configure your application using SAP BTP console client. You can configure HTTP, Mail, or RFC destinations using a standard properties file.

The tasks listed below demonstrate how to upload, download, and delete connectivity destinations. You can perform these operations for destinations related to your own subaccount, a provider subaccount, your own application, or an application provided by another subaccount.

To use an application from another subaccount, you must be subscribed to this application through your subaccount.

Note

Destination files must be encoded in **ISO 8859-1** character encoding.

Prerequisites

- You have downloaded and set up the console client. For more information, see [Set Up the Console Client](#).
- For specific information about all connectivity restrictions, see [Connectivity](#) → section "Restrictions".

HTTP Destination Properties Files

- `Name` - the name of the destination.
- `URL` - the URL of the remote system or service.
- `Authentication` - the type of authentication against the remote system or service.

The number of mandatory property keys varies depending of the authentication type you choose. For more information about HTTP destination properties files, [HTTP Destinations \[page 91\]](#).

Key stores and trust stores must be stored in JKS files with a standard `.jks` extension.

If mandatory fields are missing or data is specified incorrectly, you will be prompted accordingly by the console client.

Mail Destination Properties Files

- `Name` - the name of the destination.
- `Type` - must be "MAIL" for mail destinations.
- `mail.*` - `javax.mail` properties for configuring the mail session.

For more information about mail destination properties files, see [Mail Destinations \[page 128\]](#).

If mandatory fields are missing or data is specified incorrectly, you will be prompted accordingly by the console client.

RFC Destination Properties Files

- `Name` - the name of the destination.
- `Type` - must be "RFC" for RFC destinations.
- `jco.client*` - JCo properties for configuring an RFC connection.
- `jco.destination*` - JCo properties for configuring the behavior of a JCo destination.

All properties except `Name` and `Type` must start with "`jco.client.`" or "`jco.destination.`". For more information about RFC destination properties files, see [RFC Destinations \[page 114\]](#).

If mandatory fields are missing or data is specified incorrectly, you will be prompted accordingly by the console client.

Tasks

- [Upload Destinations \[page 58\]](#)
- [Download Destinations \[page 59\]](#)
- [Delete Destinations \[page 61\]](#)

Scenarios

- [Example: Send E-Mails \[page 212\]](#)
- [Consume Internet Services \(Java Web or Java EE 6 Web Profile\) \[page 156\]](#)
- [Consume Backend Systems \(Java Web or Java EE 6 Web Profile\) \[page 171\]](#)
- [Invoke ABAP Function Modules in On-Premise ABAP Systems \[page 195\]](#)

Related Information

[Examples \(Console\) \[page 62\]](#)

1.3.1.1.1 Upload Destinations

Context

The procedure below explains how you can upload destination configuration properties files and certificate files. You can upload them on subaccount, application or subscribed application level.

📘 Note

Bear in mind that, by default, your destinations are configured on SAP BTP, that is the hana.ondemand.com landscape. If you need to specify a particular region host, you need to add the `--host` parameter, as shown in the examples. Otherwise, you can skip this parameter.

Procedure

1. Open the command prompt.
2. Navigate to the `tools` folder of the SDK location.
3. Optional: Enter `neo help` to display all the commands of the console client or `neo help <command_name>` to display the help information for a command.
4. Upload a destination.

- To upload a destination on subaccount level, use the following command:

```
neo put-destination --account <subaccount_name> --user <user_name> --localpath <destination_or_JKS_file_localpath> --host <host>
```

- To upload a destination on application level, use the following command:

```
neo put-destination --account <subaccount_name> --user <user_name> --application <application_name> --localpath <destination_or_JKS_file_localpath> --host <host>
```

- To upload a destination for a subscribed application, use the following command:

```
neo put-destination --account <subaccount_name> --user <user_name> --application <provider_subaccount>:<provider_application> --localpath <destination_or_JKS_file_localpath> --host <host>
```

Tips

ⓘ Note

When uploading a destination configuration file that contains a password field, the password value remains available in the file. However, if you later download this file, using the `get-destination` command, the password value will no more be visible. Instead, after `Password = . . .`, you will only see an empty space.

ⓘ Note

The configuration parameters used by SAP BTP console client can be defined in a properties file as well. This may be done instead of specifying them directly in the command (with the exception of the `-password` parameter, which must be specified when the command is executed). When you use a properties file, enter the path to it as the last command line parameter.

Example:

```
neo put-destination <path_to_properties_file>
```

Related Information

[Examples \(Console\) \[page 62\]](#)

[put-destination](#)

1.3.1.1.2 Download Destinations

Context

The procedure below explains how you can download (read) destination configuration properties files and certificate files. You can download them on subaccount, application or subscribed application level.

You can read destination files with extension `.props`, `.properties`, `.jks`, and `.txt`, as well as files with no extension. Destination files must be encoded in **ISO 8859-1** character encoding.

ⓘ Note

Bear in mind that, by default, your destinations are configured on SAP BTP, that is the hana.ondemand.com landscape. If you need to specify a particular region host, you need to add the `--host` parameter, as shown in the examples. Otherwise, you can skip this parameter.

Procedure

1. Open the command prompt.
2. Navigate to the `tools` folder of the SDK location.
3. Optional: Enter `neo help` to display all the commands of the console client or `neo help <command_name>` to display the help information for a command.
4. Download a destination.
 - To read a destination on subaccount level, use the following command:

```
neo get-destination --account <subaccount_name> --user <user_name> --name <destination_name> --localpath <localpath_to_destination_or_JKS_file> --host <host>
```

- To read a destination on application level, use the following command:

```
neo get-destination --account <subaccount_name> --user <user_name> --application <application_name> --name <destination_name> --localpath <localpath_to_destination_or_JKS_file> --host <host>
```

- To read a destination for a subscribed application, use the following command:

```
neo get-destination --account <subaccount_name> --user <user_name> --application <provider_subaccount>:<provider_application> --name <destination_name> --localpath <localpath_to_destination_or_JKS_file> --host <host>
```

Tips

❗ Note

If you download a destination configuration file that contains a password field, the password value will not be visible. Instead, after `Password = . . .`, you will only see an empty space. You will need to learn the password in other ways.

❗ Note

The configuration parameters used by SAP BTP console client can be defined in a properties file as well. This may be done instead of specifying them directly in the command (with the exception of the `-password` parameter, which must be specified when the command is executed). When you use a properties file, enter the path to it as the last command line parameter. A sample *weather* properties file can be found in directory `<SDK_location>\tools\samples\connectivity`.

Example:

```
neo get-destination <path_to_properties_file>
```


Related Information

[Examples \(Console\) \[page 62\]](#)

[put-destination](#)

1.3.1.1.3 Delete Destinations

Context

The procedure below explains how you can delete destination configuration properties files and certificate files. You can delete them on subaccount, application or subscribed application level.

Note

Bear in mind that, by default, your destinations are configured on SAP BTP, that is the hana.ondemand.com landscape. If you need to specify a particular region host, you need to add the `--host` parameter, as shown in the examples. Otherwise, you can skip this parameter.

Procedure

1. Open the command prompt.
2. Navigate to the `tools` folder of the SDK location.
3. Optional: Enter `neo help` to display all the commands of the console client or `neo help <command_name>` to display the help information for a command.
4. Delete a destination.

- To delete a destination on subaccount level, use the following command:

```
neo delete-destination --account <subaccount_name> --user <user_name> --name <destination_name> --host <host>
```

- To delete a destination on application level, use the following command:

```
neo delete-destination --account <subaccount_name> --user <user_name> --application <application_name> --name <destination_name> --host <host>
```

- To delete a destination from a subscribed application, use the following command:

```
neo delete-destination --account <subaccount_name> --user <user_name> --application <provider_subaccount>:<provider_application> --name <destination_name> --host <host>
```

Tips

📌 Note

The configuration parameters used by SAP BTP console client can be defined in a properties file as well. This may be done instead of specifying them directly in the command (with the exception of the `-password` parameter, which must be specified when the command is executed). When you use a properties file, enter the path to it as the last command line parameter.

Example:

```
neo delete-destination <path_to_properties_file>
```

Related Information

[Examples \(Console\) \[page 62\]](#)

[delete-destination](#)

1.3.1.1.4 Examples (Console)

Examples for Uploading Destinations

```
neo put-destination --account mysubaccount --user p1234567890 --localpath  
C:\myfiles\myconfiguration.jks --host hanatrial.ondemand.com
```

```
neo put-destination --account mysubaccount --user p1234567890 --  
application demo --localpath C:\SDK\tools\samples\connectivity\weather --host  
hanatrial.ondemand.com
```

```
neo put-destination --account mysubaccount --user  
p1234567890 --application <provider_account>:<provider_app> --localpath  
C:\SDK\tools\samples\connectivity\weather --host hanatrial.ondemand.com
```

Examples for Downloading Destinations

```
neo get-destination --account mysubaccount --user p1234567890 --name weather --  
localpath C:\myfiles --host hanatrial.ondemand.com
```

```
neo get-destination --account mysubaccount -user p1234567890 --application demo
--name myconfiguration.jks --localpath C:\SDK\tools\samples\connectivity --host
hanatrial.ondemand.com
```

```
neo get-destination --account mysubaccount --user p1234567890 --
application <provider_account>:<provider_app> --name weather --localpath
C:\SDK\tools\samples\connectivity --host hanatrial.ondemand.com
```

Examples for Deleting Destinations

```
neo delete-destination --account mysubaccount --user p1234567890 --name
myconfiguration.jks --host hanatrial.ondemand.com
```

```
neo delete-destination --account mysubaccount --user p1234567890 --application
demo --name weather --host hanatrial.ondemand.com
```

```
neo delete-destination --account mysubaccount --user p1234567890 --application
<provider_account>:<provider_app> --name weather --host hanatrial.ondemand.com
```

1.3.1.2 Configure Destinations from the Eclipse IDE

You can use the [Connectivity](#) editor in the Eclipse IDE to configure HTTP, Mail, RFC and LDAP destinations in order to:

- Connect your cloud application to the Internet or make it consume an on-premise back-end system via HTTP(S);
- Send an e-mail from a simple Web application using an e-mail provider that is accessible on the Internet;
- Make your cloud application invoke a function module in an on-premise ABAP system via RFC.
- Use LDAP-based user authentication for your cloud application.

You can create, delete and modify destinations to use them for direct connections or export them for further usage. You can also import destinations from existing files.

Note

Destination files must be encoded in **ISO 8859-1** character encoding.

Prerequisites

- You have downloaded and set up your Eclipse IDE. For more information, see [Setting Up the Development Environment](#) or [Updating SAP BTP SDK for Neo Environment](#).
- You have created a Java EE application. For more information, see or .

Tasks

- [Create and Delete Destinations Locally \[page 64\]](#)
- [Create and Delete Destinations on the Cloud \[page 68\]](#)
- [Use Destination Certificates \(IDE\) \[page 69\]](#)
- [Import Destinations \(IDE\) \[page 71\]](#)
- [Export Destinations \(IDE\) \[page 72\]](#)

Scenarios

- [Example: Send E-Mails \[page 212\]](#)
- [Consume Internet Services \(Java Web or Java EE 6 Web Profile\) \[page 156\]](#)
- [Consume Backend Systems \(Java Web or Java EE 6 Web Profile\) \[page 171\]](#)
- [Invoke ABAP Function Modules in On-Premise ABAP Systems \[page 195\]](#)

Related Information

[Examples \(IDE\) \[page 73\]](#)




1.3.1.2.1 Create and Delete Destinations Locally

Context

The procedure below demonstrates how you can create and configure connectivity destinations (HTTP, Mail, or RFC) on a local SAP BTP server.

Procedure

1. In the context menu of the *Servers* view, choose ► *New* ► *Server* ►.
2. Expand the *SAP* node and, as a server type, choose between:
 - **Java Web Server**
 - **Java Web Tomcat 7 Server**

- [Java Web Tomcat 8 Server](#)
 - [Java EE 6 Web Profile Server](#)
3. Choose [Finish](#).
 4. A new server `<name> [Stopped]` appears on the [Servers](#) view.
 Also, a [Servers](#) folder is created and appears in the navigation tree of the Eclipse IDE. It contains configurable folders and files you can use, for example, to change your HTTP or JMX port.
 5. On the [Servers](#) view, double-click the added server to open its editor.
 6. Go to the [Connectivity](#) tab view.
 - a. In the [All Destinations](#) section, choose the  button to create a new destination.
 - b. From the dialog window, enter a name for your destination, select its type and then choose [OK](#).
 - c. In the [URL](#) field, enter the URL of the target service to which the destination should refer.
 - d. In the [Authentication](#) dropdown box, choose the authentication type required by the target service to authenticate the calls.
 - If the target service does not require authentication, choose [NoAuthentication](#).
 - If the target service requires basic authentication, choose [BasicAuthentication](#). You need to enter a user name and a password.
 - If the target service requires a client certificate authentication, choose [ClientCertificateAuthentication](#). See [Use Destination Certificates \(IDE\) \[page 69\]](#).
 - e. Optional: In the [Properties](#) or [Additional Properties](#) section, choose the  button to specify additional destination properties.
 - f. Save the editor.
 7. When a new destination is created, the changes take effect immediately.
 8. To delete a destination, choose the  button.

Related Information

[Examples \(IDE\) \[page 73\]](#)

[Managing Destinations \[page 54\]](#)

[Encrypt Destination Passwords \[page 65\]](#)

1.3.1.2.1.1 Encrypt Destination Passwords

Encrypt password fields for a destination configuration file.

When using a local server, the destination configuration is stored in the file system as plain text by default. The plain text storage includes password fields, which can be a security issue.

Perform the following procedure to encrypt those fields for a particular destination configuration file.

Generate a Key

To encrypt and decrypt the password fields, you need a key for an AES-128-CBC algorithm (Advanced Encryption Standard). The following steps show you how to generate this key using OpenSSL. Alternatively, you can use any other appropriate procedure.

Note

If a stronger AES algorithm is required (for example, AES with 256-bit keys), you must install the [JCE Unlimited Strength Jurisdiction Policy Files](#) in the JDK/JRE.

Prerequisites

OpenSSL is provided by Linux and Mac by default. For Windows, you must install it from <http://gnuwin32.sourceforge.net/packages/openssl.htm>.

Note

For Windows, the installer does not add the path of the `openssl.exe` file to the PATH environment variable. You should do this manually or navigate to the file before executing the OpenSSL commands in the terminal.

Procedure

1. Open a terminal.
2. (Optional) Navigate to the OpenSSL executable.
3. Adjust and execute the following command:

```
openssl enc -aes-128-cbc -P -k <pass_phrase> > <folder>\<name_of_the_key>
```

where `<pass_phrase>` is a sequence of characters, based on which the key is generated.

Sample Code

```
openssl enc -aes-128-cbc -P -k "my test pass phrase" > "E:\mykeys\Key1"
```

4. This procedure generates a key and stores it in the specified file (and creates the file if necessary). The key file has the following format:

```
salt=3F190F676A469E24
key=C9BA8910B87D25242AF759001842EFCF
iv =AD5EE334AE9694BE96E1754B6E736C7D
```

Note

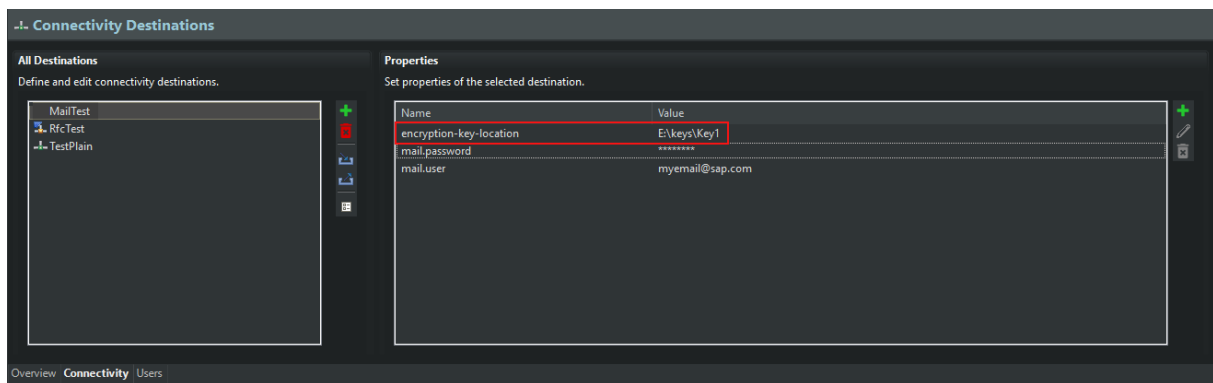
Only the `<key>` and `<iv>` fields are needed. If you use a different method to create the key file, you only need to include those two fields.

Configure Encryption

To store the password fields of a destination in an encrypted format, you must set the `encryption-key-location` property. The value of this property is the absolute path of the key file, containing an encryption key in the format described above.

Note

You should store the key file on a removable storage device. Otherwise, the decryption key can always be accessed.



When you save the destination, the destination file in the file system includes encrypted password fields. The key location, which is specified by the `encryption-key-location` property, is used when a destination is retrieved by an application to get the key and encrypt the password fields. This is done automatically by the service.

Encryption/Decryption Failure

Destination Editor (Eclipse)

- *Encryption*
Encryption is performed when the destination is saved to the file system. If an error occurs, the Save operation fails and a message shows the cause.
- *Decryption*
The following error cases may occur. If:
 - a key file is missing in the file system, the editor lets you edit the destination and specify a new location of the key.

Note

The Save operation fails until a valid key (which can decrypt the loaded destination) is provided. We strongly recommend that you provide the new location of the key immediately and save the destination. Then you can continue working with the destination as usual.

- a key file is corrupted, the editor treats it as if the key was not found. You can specify a new location and, if the key is valid, continue working with the destination.
- a particular field (or multiple fields) cannot be decrypted, the editor loads the destination and changes the value of the failed properties to **blank**. In this case, you must modify (specify new values) or remove each of these fields to fix the corrupted data.
- the initialization of the decrypting library fails, all password fields are changed to **blank**.

SDK

- *Decryption*
If decryption fails, the retrieval of an encrypted destination always causes an exception, no matter the cause of the failure. This exception is either *IllegalStateException* (if the failure is caused by a Java problem), or *IllegalArgumentException* (if the failure is caused by a problem in the destination or key file).

Note

The SDK does not perform any destination encryption.

1.3.1.2.2 Create and Delete Destinations on the Cloud

Context

The procedure below demonstrates how you can create and configure connectivity destinations (HTTP, Mail, or RFC) on SAP BTP.


Procedure

1. In the context menu of the *Servers* view, choose **New > Server**.
2. Choose *SAP Cloud Platform* as the type of server you want to create, choose *Next*, and then *Finish*.
3. A new server *<application>.<subaccount> [Stopped]* appears on the *Servers* view.
4. Double-click the added server to open the server editor.
5. Go to the *Connectivity* tab view.
 - a. In the *All Destinations* section, choose the **+** button to create a new destination.
 - b. From the dialog window, enter a name for your destination, select its type and then choose *OK*.
 - c. In the *URL* field, enter the URL of the target service to which the destination should refer.
 - d. In the *Authentication* dropdown box, choose the authentication type required by the target service to authenticate the calls.
 - If the target service does not require authentication, choose *NoAuthentication*.

- If the target service requires basic authentication, choose *BasicAuthentication*. You need to enter a user name and a password.
 - If the target service requires a client certificate authentication, choose *ClientCertificateAuthentication*. See [Use Destination Certificates \(IDE\) \[page 69\]](#).
 - If the target service requires your cloud user authentication, choose *PrincipalPropagation*. You also need to select *Proxy Type: OnPremise* and should enter the additional property *CloudConnectorVersion* with value **2**.
- e. In the *Proxy Type* dropdown box, choose the required type of proxy connection.

Note

This dropdown box allows you to choose the type of your proxy and is only available when deploying on SAP BTP. The default value is *Internet*. In this case, the destination uses the HTTP proxy for the outbound communication with the Internet. For consumption of an on-premise target service, choose the *OnPremise* option so that the proxy to the SSL tunnel is chosen and the tunnel is established to the connected Cloud Connector.

- f. Optional: In the *Properties* or *Additional Properties* section, choose the  button to specify additional destination properties.
- g. Save the editor. This saves the specified destination configuration in SAP BTP.
6. When new destinations are created, the changes take effect immediately.

Note

Bear in mind that changes are currently cached with a cache expiration of up to 4 minutes. That's why if you modify a destination configuration, the changes might not take effect immediately. However, if the relevant Web application is restarted on the cloud, the destination changes will take effect immediately.

7. To delete a destination, choose the  button.

Related Information

[Examples \(IDE\) \[page 73\]](#)

[Managing Destinations \[page 54\]](#)

1.3.1.2.3 Use Destination Certificates (IDE)

Prerequisites

You have opened the *Connectivity* editor in the Eclipse IDE.


Context

You can maintain keystore certificates in the [Connectivity](#) editor. You can upload, add and delete certificates for your connectivity destinations. Bear in mind that:

- You can use JKS, PFX and P12 files for destination keystore, and JKS, CRT, CER, DER files for destination truststore.
- You add certificates in a keystore file and then you upload, add, or delete this keystore.
- You can add certificates only for **HTTPS** destinations. Keystore is available only for `ClientCertificateAuthentication`.


Procedure

Uploading Certificates

1. Press the [Upload/Delete keystore](#)  button. You can find it in the [All Destinations](#) section in the [Connectivity](#) editor.
2. Choose [Upload Keystore](#) and select the certificate you want to upload. Choose [Open](#) or double-click the certificate.

The certificate file is added.

Note

You can upload a certificate during creation or editing of a destination, by choosing [Manage Keystore](#) or by pressing the [Upload/Delete keystore](#)  button.

Adding Certificates to Destinations

1. Create a new destination, or open an existing one for editing.
2. In the [URL](#) field, enter an HTTPS address.
3. You can use the default JDK truststore or select another one from the truststore dropdown menu. If the menu is empty, you can upload a certificate on the fly. The password is used for the keystore that will contain your certificate on the cloud.
4. In the [Authentication](#) field, select [ClientCertificateAuthentication](#).
5. In the [Keystore name](#) field, select the certificate you just added. Enter the password.

Deleting Certificates

1. Press the [Upload/Delete keystore](#)  button.

2. Select the certificate you want to remove and choose [Delete Selected](#)
3. Upload another certificate, or close the [Manage Keystores](#) window.

Related Information

[Create and Delete Destinations Locally \[page 64\]](#)

[Create and Delete Destinations on the Cloud \[page 68\]](#)

[Import Destinations \(IDE\) \[page 71\]](#)

1.3.1.2.4 Import Destinations (IDE)


Prerequisites

You have previously created a connectivity destination (HTTP, Mail, or RFC).

Note

The [Connectivity](#) editor allows importing destination files with extension **.props**, **.properties**, and **.txt**, as well as files with no extension. Destination files must be encoded in **ISO 8859-1** character encoding.

Procedure

1. On the [Servers](#) view, double-click your server to open its editor.
2. Go to the [Connectivity](#) tab view.
3. Choose button  ([Import destination](#)).
4. Browse to one of the following file types and choose [OK](#).
 - If the destination does not contain client certificate authentication, it is saved as a single configuration file.
 - If the destination provides client certificate data, it is saved as an archive, which contains the main configuration file and a Keystore file.
5. The destination file is imported within the [Connectivity](#) editor.

Note

If the properties file contains incorrect properties or values, for example wrong destination type, the editor only displays the valid ones in the [Properties](#) table.

Related Information


[Examples \(IDE\) \[page 73\]](#)

1.3.1.2.5 Export Destinations (IDE)

Prerequisites

You have imported or created a new destination (HTTP, Mail, or RFC) in the Eclipse IDE.

Procedure

1. On the [Servers](#) view, double-click your server to open its editor.
2. Go to the [Connectivity](#) tab view.
3. From the list of destination names, select the one you want to export.
4. Choose button  ([Export destination](#)).
5. Browse to the directory you want to export your destination.
 - If the destination does not contain client certificate authentication, it is saved as a single configuration file.
 - If the destination provides client certificate data, it is saved as an archive, which contains the main configuration file and a Keystore file.

→ Tip

You can keep the default name of the destination, or rename it to avoid overriding with previous files with the same name.

Next Steps

After exporting the destination, you can open it to check its content. Bear in mind that all password fields will be commented (with # symbols), and their values - deleted.

Example:

```
#Exported connectivity destination
#The following fields with passwords were removed:
#Password
#Tue Apr 21 15:01:02 FET 2015
```

```
Type=HTTP
Authentication=BasicAuthentication
Name=mydestination
URL=https://sap.com/index.html
User=p1234567890
```

Related Information

[Examples \(IDE\) \[page 73\]](#)

1.3.1.2.6 Examples (IDE)

Example of HTTP Destination (Internet)

All Destinations
Define and edit connectivity destinations.

...internetdest

Basic Properties

Set the basic properties of the selected HTTP destination.

URL:

Authentication:

Proxy Type:

Additional Properties

Set additional properties of the selected HTTP destination.

Name	Value

Overview

Connectivity

Advanced

All Destinations

Define and edit connectivity destinations.

...trusteddest

Basic Properties

Set the basic properties of the selected HTTP destination.

URL:

Authentication: NoAuthentication

Proxy Type: Internet

☐ Use default JDK truststore

Truststore file: KeyStore1.jks Manage Truststores

Password:

Additional Properties

Set additional properties of the selected HTTP destination.

Name	Value

Overview

Connectivity

Advanced

All Destinations

Define and edit connectivity destinations.

...trusteddest

Basic Properties

Set the basic properties of the selected HTTP destination.

URL:

Authentication: NoAuthentication

Proxy Type: Internet

☒ Use default JDK truststore

Additional Properties

Set additional properties of the selected HTTP destination.

Name	Value

Overview

Connectivity

Advanced

Example of HTTP Destination (OnPremise)

All Destinations
Define and edit connectivity destinations.

///...onpremiested

+

✖

Basic Properties
Set the basic properties of the selected HTTP destination.

URL:

Authentication:

Proxy Type:

Additional Properties
Set additional properties of the selected HTTP destination.

Name	Value
CloudConnectorVersion	2

+

✖

Overview

Connectivity

Advanced

Example of Mail Destination

All Destinations
Define and edit connectivity destinations.

///...weather

✉ Session

+

✖

Properties
Set properties of the selected destination.

Name	Value
mail.password	*****
mail.smtp.auth	true
mail.smtp.host	smtp.gmail.com
mail.smtp.port	587
mail.smtp.starttls.ena...	true
mail.transport.protocol	smtp
mail.user	johnsmith@googlemail.com

+

✖

Overview

Connectivity

Advanced

Example of RFC Destination

The screenshot displays the 'Destinations' editor in SAP BTP Cockpit. It is divided into two main sections: 'All Destinations' and 'Properties'.

All Destinations: This section contains a list of destinations. A single destination named 'JCoSystem' is listed. To the right of the list are icons for adding (+), deleting (X), cloning (fork), and importing/exporting (document with arrows).

Properties: This section is titled 'Set properties of the selected destination.' It contains a table with the following data:

Name	Value
jco.client.ashost	abapserver.hana.cloud
jco.client.client	000
jco.client.lang	EN
jco.client.passwd	*****
jco.client.sysnr	42
jco.client.user	MYJCO
jco.destination.pea...	10
jco.destination.pool...	5

At the bottom of the editor, there are three tabs: 'Overview', 'Connectivity', and 'Advanced'. The 'Connectivity' tab is currently selected.

1.3.1.3 Configure Destinations from the Cockpit

Use the [Destinations](#) editor in SAP BTP cockpit to configure HTTP, Mail, RFC, and LDAP destinations in order to:

- Connect your cloud application to the Internet or make it consume an on-premise back-end system via HTTP(S).
- Send an e-mail from a simple Web application using an e-mail provider that is accessible on the Internet.
- Make your cloud application invoke a function module in an on-premise ABAP system via RFC.
- Use LDAP-based user authentication for your cloud application.

You can create, delete, clone, modify, import and export destinations.

Use this editor to work with destinations on subscription, subaccount, and application level.

📌 Note

Destination files must be encoded in **ISO 8859-1** character encoding.

Prerequisites

1. You have logged into the cockpit from the SAP BTP landing page, depending on your subaccount type. For more information, see [Regions and Hosts Available for the Neo Environment](#).
2. Depending on the level you need to make destination configurations from the [Destinations](#) editor, make sure the following is fulfilled:
 - Subscription level – you need to have at least one application subscribed to your subaccount.
 - Application level – you need to have at least one application deployed on your subaccount.
 - Subaccount level – no prerequisites.

For more information, see [Access the Destinations Editor \(Neo Environment\)](#) [page 78].

Tasks

- [Create Destinations \(Cockpit\)](#) [page 79]
- [Check the Availability of a Destination \(Cockpit\)](#) [page 83]
- [Import Destinations \(Cockpit\)](#) [page 87]
- [Clone Destinations \(Cockpit\)](#) [page 84]
- [Export Destinations \(Cockpit\)](#) [page 88]
- [Edit and Delete Destinations \(Cockpit\)](#) [page 85]

Scenarios

- [Example: Send E-Mails](#) [page 212]
- [Consume Internet Services \(Java Web or Java EE 6 Web Profile\)](#) [page 156]
- [Consume Backend Systems \(Java Web or Java EE 6 Web Profile\)](#) [page 171]
- [Invoke ABAP Function Modules in On-Premise ABAP Systems](#) [page 195]

Related Information

[Examples \(Cockpit\)](#) [page 89]

1.3.1.3.1 Access the Destinations Editor (Neo Environment)

Prerequisites

- You have logged into the cockpit from the SAP BTP landing page, depending on your global account type. For more information, see [Regions and Hosts Available for the Neo Environment](#).

Procedure

Access on Subscription Level

1. In the cockpit, select your subaccount name from the [Subaccount](#) menu in the breadcrumbs.
2. From the left-side navigation, choose ► [Applications](#) ► [Subscriptions](#) ▾ to open the page with your currently subscribed Java applications (if any).
3. Select the application for which you need to create a destination.
4. From the left-side panel, choose [Destinations](#).

Access on Subaccount Level

1. In the cockpit, select your subaccount name from the [Subaccount](#) menu in the breadcrumbs.
2. From the left-side navigation, choose ► [Connectivity](#) ► [Destinations](#) ▾.
3. The [Destinations](#) editor is opened.

Access on Application Level

1. In the cockpit, select your subaccount name from the [Subaccount](#) menu in the breadcrumbs.
2. From the left-side navigation, choose ► [Applications](#) ► [Java Applications](#) ▾ to open the page with your currently deployed Java Web applications (if any).
3. Select the application for which you need to create a destination.
4. From the left-side panel, choose ► [Configuration](#) ► [Destinations](#) ▾.
5. The [Destinations](#) editor is opened.

Related Information

[Create Destinations \(Cockpit\) \[page 79\]](#)

[Import Destinations \(Cockpit\) \[page 87\]](#)

[Edit and Delete Destinations \(Cockpit\) \[page 85\]](#)

1.3.1.3.2 Create Destinations (Cockpit)

Prerequisites

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

Context

To learn how to create HTTP, RFC, and Mail destinations, follow the steps on the relevant pages:

- [Create HTTP Destinations \[page 79\]](#)
- [Create RFC Destinations \[page 81\]](#)
- [Create Mail Destinations \[page 82\]](#)

Related Information

[Managing Destinations \[page 54\]](#)

[Examples \(Cockpit\) \[page 89\]](#)

1.3.1.3.2.1 Create HTTP Destinations

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

Prerequisites

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

Procedure

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


Note

For more information, see also: [HTTP Destinations \[page 91\]](#).

7. From the [Authentication](#) dropdown box, select the authentication type you need for the connection.
For details, see [HTTP Destinations \[page 91\]](#).

Note

If you set an **HTTPS** destination, you need to also add a Trust Store. For more information, see [Use Destination Certificates \(Cockpit\) \[page 86\]](#).

8. (Optional) If you are using more than one Cloud Connector for your subaccount, you must enter the [Location ID](#) of the target Cloud Connector.
See also [Managing Subaccounts \[page 291\]](#) (section **Procedure**, step 4).
9. (Optional) You can enter additional properties.
 - a. In the [Additional Properties](#) panel, choose [New Property](#).
 - b. Enter a key (name) or choose one from the dropdown menu and specify a value for the property. You can add as many properties as you need.
 - c. To delete a property, choose the  button next to it.

Note

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

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

Related Information

[Examples \(Cockpit\) \[page 89\]](#)

[HTTP Destinations \[page 91\]](#)

[Edit and Delete Destinations \(Cockpit\) \[page 85\]](#)

1.3.1.3.2.2 Create RFC Destinations

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

Prerequisites


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

Procedure

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

Note

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

6. Enter credentials for [User](#) and [Password](#).
7. (Optional) Enter an [Alias User](#) name. See also [User Logon Properties \[page 114\]](#).
8. (Optional) Enter credentials for [Repository User](#) and [Repository Password](#), if you want to use a different user for repository lookups. See also [Repository Configuration \[page 118\]](#).
9. (Optional) If you are using more than one Cloud Connector for your subaccount, you must enter the [Location ID](#) of the target Cloud Connector.
See also [Managing Subaccounts \[page 291\]](#) (section **Procedure**, step 4).
10. (Optional) You can enter additional properties.
 - a. In the [Additional Properties](#) panel, choose [New Property](#).
 - b. Enter a key (name) or choose one from the dropdown menu and specify a value for the property. You can add as many properties as you need.
 - c. To delete a property, choose the  button next to it.

If you add `PrincipalPropagation` as additional property (`jco.destination.auth_type`), your RFC destination must not contain user and password information. It can, however, contain repository credentials.

Note

For a detailed description of RFC-specific properties (JCo properties), see [RFC Destinations \[page 114\]](#).

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

Related Information

[Examples \(Cockpit\) \[page 89\]](#)

[RFC Destinations \[page 114\]](#)

[Edit and Delete Destinations \(Cockpit\) \[page 85\]](#)


[Cloud Connector \(Neo Environment\) \[page 232\]](#)

1.3.1.3.2.3 Create Mail Destinations

Prerequisites

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

Procedure

1. Choose [New Destination](#).
2. Enter a destination name.
3. From the [Type](#) dropdown menu, choose **MAIL**.
4. The [Description](#) field is optional.
5. Enter credentials for [User](#) and [Password](#).
6. Optional: You can enter additional properties.
 - a. In the [Additional Properties](#) panel, choose [New Property](#).
 - b. Enter a key (name) or choose one from the dropdown menu and specify a value for the property. You can add as many properties as you need.
 - c. To delete a property, choose the  button next to it.
7. When you are ready, choose the [Save](#) button.

Related Information

[Examples \(Cockpit\) \[page 89\]](#)

[Mail Destinations \[page 128\]](#)

[Edit and Delete Destinations \(Cockpit\) \[page 85\]](#)

1.3.1.3.3 Check the Availability of a Destination (Cockpit)

Prerequisites

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

Context

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

Note

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

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

Note

The check does not guarantee that a backend is operational. It only verifies if a connection to the backend is possible.

This check is supported only for destinations with [Proxy Type Internet](#) and [OnPremise](#):

- For [Internet](#) destinations:
 - If the check receives an **HTTP status code above or equal to 500** from the targeted URL, the check is considered failed.
 - Every HTTP status code below 500 is treated as successful.
- For [OnPremise](#) destinations:
 - If the targeted backend is reached and returns an **HTTP status code below 500** the check is considered successful.

Error Messages for OnPremise Destinations


Error Message	Reason	Action
<i>Backend status could not be determined.</i>	<ul style="list-style-type: none"> The Cloud Connector version is less than 2.7.1. The Cloud Connector is not connected to the subaccount. The backend returns a HTTP status code above or equal to 500 (server error). The Cloud Connector is not configured properly. 	<ul style="list-style-type: none"> Upgrade the Cloud Connector to version 2.7.1 or higher. Connect the Cloud Connector to the corresponding subaccount. Check the server status (availability) of the back-end system. Check the basic Cloud Connector configuration steps: Initial Configuration [page 278]
<i>Backend is not available in the list of defined system mappings in Cloud Connector.</i>	The Cloud Connector is not configured properly.	Check the basic Cloud Connector configuration steps: Initial Configuration [page 278]
<i>Resource is not accessible in Cloud Connector or backend is not reachable.</i>	The Cloud Connector is not configured properly.	Check the basic Cloud Connector configuration steps: Initial Configuration [page 278]
<i>Backend is not reachable from Cloud Connector.</i>	Cloud connector configuration is ok but the backend is not reachable.	Check the backend (server) availability.

1.3.1.3.4 Clone Destinations (Cockpit)

Prerequisites

You have previously created or imported a connectivity destination (HTTP, Mail, or RFC) in the [Destinations](#) editor of the cockpit.

Procedure

1. In the [Destinations](#) editor, go to the existing destination which you want to clone.
2. Choose the  icon.
3. The editor automatically creates and opens a new destination that contains all the properties of the selected one.
4. You can modify some parameters if you need.

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

Related Information

[Examples \(Cockpit\) \[page 89\]](#)


[Export Destinations \(Cockpit\) \[page 88\]](#)

1.3.1.3.5 Edit and Delete Destinations (Cockpit)

Prerequisites

You have previously created or imported a connectivity destination (HTTP, Mail, or RFC) in the [Destinations](#) editor of the cockpit.

Procedure

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

→ Tip

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

- Delete a destination:

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

Related Information

[Examples \(Cockpit\) \[page 89\]](#)

[Export Destinations \(Cockpit\) \[page 88\]](#)

1.3.1.3.6 Use Destination Certificates (Cockpit)

Prerequisites

You have logged into the cockpit and opened the [Destinations](#) editor. For more information, see [Access the Destinations Editor \(Neo Environment\) \[page 78\]](#).

Context

⚠ Caution

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

This page explains how you can maintain truststore and keystore certificates in the [Destinations](#) editor. You can upload, add and delete certificates for your connectivity destinations. Bear in mind that:

- You can only use JKS, PFX and P12 files for destination key store, and JKS, CRT, CER, DER for destination trust store.
- You can add certificates only for **HTTPS** destinations. Truststore can be used for all authentication types. Keystore is available only for `ClientCertificateAuthentication`.
- An uploaded certificate file should contain the entire certificate chain.

Procedure

Uploading Certificates

1. Choose the [Certificates](#) button.
2. Choose [Upload Certificate](#).
3. Browse to the certificate file you need to upload.
The certificate file is added.

📘 Note

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

Adding Certificates to Destinations

1. Create a new destination, or open an existing one for editing.
2. In the *URL* field, enter an HTTPS address.
3. You can use the default JDK truststore or select another one from the dropdown menu. If the menu is empty, you can upload a certificate on the fly. To omit this property, you can set `TrustAll=true`.
4. If you choose *Authentication* = **ClientCertificateAuthentication**, you need to also provide a keystore.

Deleting Certificates

1. Choose the *Certificates* button or click the *Upload and Delete Certificates* link.
2. Select the certificate you want to remove and choose *Delete Selected*.
3. Upload another certificate, or close the *Certificates* window.

Related Information

[Create Destinations \(Cockpit\) \[page 79\]](#)

[Import Destinations \(Cockpit\) \[page 87\]](#)

[Edit and Delete Destinations \(Cockpit\) \[page 85\]](#)

1.3.1.3.7 Import Destinations (Cockpit)

Prerequisites

You have previously created a connectivity destination (HTTP, Mail, or RFC).

Note

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

Procedure

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

Related Information

[Examples \(Cockpit\) \[page 89\]](#)

[Edit and Delete Destinations \(Cockpit\) \[page 85\]](#)


1.3.1.3.8 Export Destinations (Cockpit)

Export destinations from the [Destinations](#) editor in the SAP BTP cockpit to backup or reuse a destination configuration.

Prerequisites

You have created a connectivity destination (HTTP, Mail, or RFC) in the [Destinations](#) editor.

Procedure

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

Related Information

[Examples \(Cockpit\) \[page 89\]](#)

[Edit and Delete Destinations \(Cockpit\) \[page 85\]](#)

1.3.1.3.9 Examples (Cockpit)

Example of HTTP Destination (Internet, Client Certificate Authentication)

The screenshot shows the 'Destination Configuration' form in SAP Cockpit. The form is for a new destination named 'MyInternet'. The configuration details are as follows:

- Name:** MyInternet
- Type:** HTTP
- Description:** (empty)
- URL:** https://google.com
- Proxy Type:** Internet
- Authentication:** ClientCertificateAuthentication
- Key Store Location:** MyKeyStore.jks
- Key Store Password:** (masked with asterisks)

Additional Properties:

- Trust Store Location:** MyTrustStore.jks
- Trust Store Password:** (empty)

Buttons: Save, Cancel, New Property, Upload and Delete Certificates.

Example of HTTP Destination (Internet, OAuth2SAMLBearerAssertion)

The screenshot shows the 'Destination Configuration' form in SAP Cockpit for a destination named 'MyDestination'. The configuration details are as follows:

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

Additional Properties:

- Use default JDK truststore:** (checked)

Buttons: Save, Cancel, New Property, Upload and Delete Certificates.

Example of HTTP Destination (On-Premise)

The screenshot shows the 'Destination Configuration' window for an HTTP destination. The 'Name' is 'MyOnPremiseDestination', 'Type' is 'HTTP', 'Description' is 'test', 'Location ID' is '<optional>', 'URL' is 'https://mycompany.corp:443', 'Proxy Type' is 'OnPremise', 'Authentication' is 'BasicAuthentication', 'User' is 'SomeUser', and 'Password' is masked with asterisks. The 'Additional Properties' section is empty. The 'Save' and 'Cancel' buttons are at the bottom left.

Example of Mail Destination

The screenshot shows the 'Destination Configuration' window for a Mail destination. The 'Name' is 'MyMailDestination', 'Type' is 'MAIL', 'Description' is empty, 'User' is 'john.smith@gmail.com', and 'Password' is masked with asterisks. The 'Additional Properties' section contains four properties: 'mail.smtp.yyyy' (true), 'mail.smtp.auth' (true), 'mail.smtp.host' (smtp.gmail.com), and 'mail.smtp.port' (587). The 'Save' and 'Cancel' buttons are at the bottom left.

Example of RFC Destination

The screenshot shows the 'Destination Configuration' window for an RFC destination. The 'Name' is 'MyJCoSystem', 'Type' is 'RFC', 'Description' is empty, 'Location ID' is empty, 'User' is 'MyJCoUser', 'Password' is masked with asterisks, 'Repository User' is empty, and 'Repository Password' is empty. The 'Additional Properties' section contains four properties: 'jco.client.lang' (EN), 'jco.client.ashost' (abapserver.hana.cloud), 'jco.client.client' (000), and 'jco.client.sysnr' (42). The 'Save' and 'Cancel' buttons are at the bottom left.

The following main properties correspond to the relevant additional properties:

User → `jco.client.user`

Password → `jco.client.passwd`

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

Note

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

Related Information

[HTTP Destinations \[page 91\]](#)

[RFC Destinations \[page 114\]](#)

[Mail Destinations \[page 128\]](#)

1.3.2 HTTP Destinations

Overview

The HTTP destinations provide data communication via HTTP protocol and are used for both Internet and on-premise connections.

HTTP Destination Properties

The runtime tries to resolve a destination in the order: *Subscription Level* → *Subaccount Level* → *Application Level*. By using the optional "DestinationProvider" property, a destination can be limited to application level only, that is, the runtime tries to resolve the destination on application level.

Property	Description
<code>DestinationProvider</code>	Restricts destination to application level. If the property is specified, the destination will be searched on the application level only. By default, destinations are searched on all configuration levels.
<code>URL.connectionTimeoutInSeconds</code>	Period of time in which the HTTP client expects a confirmation after it initiated a new TCP connection. Introduced in version 2.15.
<code>URL.socketReadTimeoutInSeconds</code>	Period of time the HTTP client will wait for receiving a response (data) from the server. Introduced in version 2.15.

Note

If you use Java Web Tomcat 7 runtime container, the `DestinationProvider` property is not supported. Instead, you can use [AuthenticationHeaderProvider API \[page 142\]](#).

Example

```
Name=weather
Type=HTTP
Authentication=NoAuthentication
DestinationProvider=Application
```

Supported Proxy Types for Connectivity

The proxy types supported by the Connectivity service are:

- [Internet](#) - The application can connect to an external REST or SOAP service on the Internet.
- [OnPremise](#) - The application can connect to an on-premise back-end system through the Cloud Connector.

The proxy type used for a destination must be specified by the destination property `ProxyType`. The property's default value (if not configured explicitly) is [Internet](#).

Setting Proxy Types for SAP HANA Cloud Local Runtime

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

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

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

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

For more information and example, see [Consume Internet Services \(Java Web or Java EE 6 Web Profile\) \[page 156\]](#).

Setting Proxy Types for SAP HANA Cloud

- When using the [Internet](#) proxy type, you do not need to perform any additional configuration steps.
- When using the [OnPremise](#) proxy type, you configure the setting the standard way through the [Connectivity](#) editor in the Eclipse IDE.
For more information and example, see [Consume Backend Systems \(Java Web or Java EE 6 Web Profile\)](#) [page 171].

Configuring Authentication

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

- [Server Certificate Authentication](#) [page 93]
- [SAP Assertion SSO Authentication](#) [page 96]
- [Principal Propagation SSO Authentication for HTTP](#) [page 99]
- [SAML Bearer Assertion Authentication](#) [page 101]
- [Application-to-Application SSO Authentication](#) [page 104]
- [Client Authentication Types for HTTP Destinations](#) [page 108]
- [OAuth Client Credentials Authentication](#) [page 110]
- [OAuth with X.509 Client Certificates](#) [page 113]

1.3.2.1 Server Certificate Authentication

Context

The server certificate authentication is applicable for all client authentication types, described below.

Note

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

Properties

Property	Description
TLSVersion	Optional property. Can be used to specify the preferred TLS version to be used by the current destination. Since TLS 1.2 is not enabled by default on the older java versions this property can be used to configure TLS 1.2 in case this is required by the server configured in this destination. It is usable only in HTTP destinations. Example: TLSVersion= TLSv1.2 .
TrustStoreLocation	Path to the JKS file which contains trusted certificates (Certificate Authorities) for authentication against a remote client. <ol style="list-style-type: none"> When used in local environment When used in cloud environment <ol style="list-style-type: none"> The relative path to the JKS file. The root path is the server's location on the file system. The name of the JKS file. <div> <p>Note</p> <p>The default JDK trust store is appended to the trust store defined in the destination configuration. As a result, the destination simultaneously uses both trust stores. If the TrustStoreLocation property is not specified, the JDK trust store is used as a default trust store for the destination.</p> </div>
TrustStorePassword	Password for the JKS trust store file. This property is mandatory if TrustStoreLocation is used.
TrustAll	<p>If this property is set to TRUE in the destination, the server certificate will not be checked for SSL connections. It is intended for test scenarios only, and should not be used in production (since the SSL server certificate is not checked, the server is not authenticated). The possible values are TRUE or FALSE; the default value is FALSE (that is, if the property is not present at all).</p> <p>In case TrustAll = TRUE, the TrustStoreLocation property is ignored so you can omit it.</p> <p>In case <TrustAll> = FALSE, the <TrustStoreLocation> property is mandatory to be used.</p>

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

Note

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

Note

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

Configuration

- [Configure Destinations from the Cockpit \[page 76\]](#)
- [Configure Destinations from the Eclipse IDE \[page 63\]](#)
- [Configure Destinations from the Console Client \[page 55\]](#)

Related Information

[Client Authentication Types for HTTP Destinations \[page 108\]](#)

1.3.2.2 SAP Assertion SSO Authentication

Create and configure an SAP Assertion SSO destination for an application in the Neo environment.

⚠ Caution

Authentication type SAP Assertion SSO is deprecated. Use [Principal Propagation SSO Authentication for HTTP \[page 99\]](#) instead, which is the recommended mechanism for establishing single sign-on (SSO).

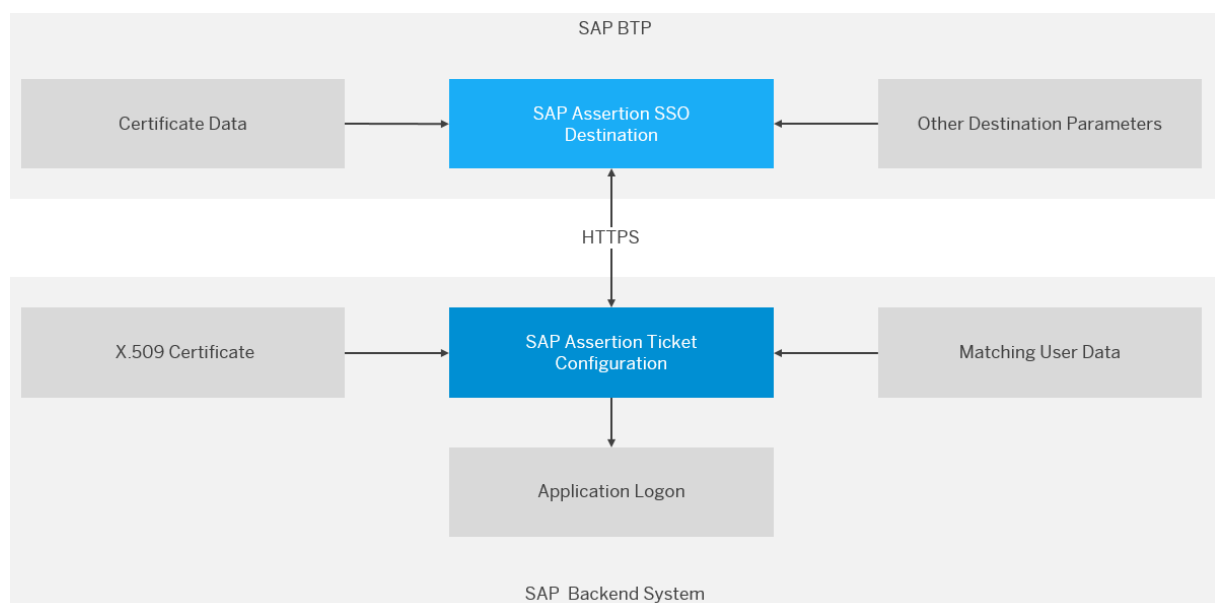
Context

By default, all SAP systems accept SAP assertion tickets for user propagation.

📘 Note

For more information, see [Authentication Assertion Tickets](#).

The aim of the [SAPAssertionSSO](#) destination is to generate such an assertion ticket in order to propagate the currently logged-on SAP BTP user to an SAP backend system. You can only use this authentication type if the user IDs on both sides are the same. The following diagram shows the elements of the configuration process on the SAP BTP and in the corresponding backend system:



Configuration Steps

1. Configure the back-end system so that it can accept SAP assertion tickets signed by a trusted x.509 key pair. For more information, see [Configuring a Trust Relationship for SAP Assertion Tickets](#).
2. Create and configure a *SAPAssertionSSO* destination by using the properties listed below, and deploy it on SAP BTP.
 - [Configure Destinations from the Cockpit \[page 76\]](#)
 - [Configure Destinations from the Console Client \[page 55\]](#)

Note

Configuring *SAPAssertionSSO* destinations from the Eclipse IDE is not yet supported.

Properties

The following credentials need to be specified:

Property	Description
Name	Destination name. It must be the same as the destination name you use for the configuration tools, that is, the console client and <i>Destinations</i> editor (cockpit).
Type	Destination type. Use HTTP for all HTTP(S) destination.
URL	URL of the protected resource on the called application
Authentication	Authentication type. Use SAPAssertionSSO as a value.
IssuerSID	This system ID should be trusted by the back-end system.
IssuerClient	This client ID should be trusted by the back-end system.
RecipientSID	System ID (SID) of the back-end system
RecipientClient	Client ID of the back-end system
Certificate	A base64 encoded certificate that is trusted by the SAP system
SigningKey	A base64 encoded signing/private key that is trusted by the SAP system
SystemUser	Optional property. <ul style="list-style-type: none">• If specified, all SAP assertion tickets are generated with the specified user ID.• If not specified, all SAP assertion tickets are sent on behalf of the currently logged-on user. Thus, if the current user needs to be propagated, do not use this property.
ProxyType	You can use both proxy types Internet and OnPremise .

Property	Description
CloudConnectorLocationId	<p>Optional property.</p> <p>As of Cloud Connector version 2.9.0, you can connect multiple Cloud Connectors to an account as long as their location ID is different. The value defines the location ID identifying the Cloud Connector over which the connection is opened. The default value is an empty string identifying the Cloud Connector that is connected without any location ID, which is also the case for all Cloud Connector versions prior to 2.9.0.</p>
URL.headers.<header-key>	<p>Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a URL.headers prefix separated by dot-delimiter. For example:</p> <div data-bbox="820 889 1043 929"> <p>↔ Sample Code</p> </div> <pre>URL.headers.<header-key-1>=<header-value-1> ... URL.headers.<header-key-n>=<header-value-n></pre> <div data-bbox="820 1167 928 1205"> <p>📌 Note</p> </div> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention. The HttpDestination (Version 2) [page 150] library supports this as of version 2.14.</p>

Example

```
Name=weather
Type=HTTP
Authentication=SAPAssertionSSO
IssuerSID=JAV
IssuerClient=000
RecipientSID=SAP
RecipientClient=100
Certificate=MIICiDCCAkegAwI...rvHTQ\=\=
SigningKey=MIIBSwIB...RuqNKGa\=
```

1.3.2.3 Principal Propagation SSO Authentication for HTTP

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

Context

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

Note

This authentication type applies only for on-premise connectivity.

Configuration Steps

You can create and configure a *PrincipalPropagation* destination by using the properties listed below, and deploy it on SAP BTP. For more information, see:

- [Configure Destinations from the Cockpit \[page 76\]](#)
- [Configure Destinations from the Eclipse IDE \[page 63\]](#)
- [Configure Destinations from the Console Client \[page 55\]](#)

Properties

The following credentials need to be specified:

Property	Description
Name	Destination name. It must be the same as the destination name you use in the configuration tools, that is, <i>Connectivity</i> editor (Eclipse IDE), <i>Destinations</i> editor (cockpit), and the console client.
Type	Destination type. Use HTTP for all HTTP(S) destinations.
URL	Virtual URL of the protected on-premise application.
Authentication	Authentication type. Use PrincipalPropagation as value.
ProxyType	You can only use proxy type OnPremise .
CloudConnectorLocationId	As of Cloud Connector 2.9.0, you can connect multiple Cloud Connectors to a subaccount as long as their location ID is different. The location ID specifies the

Property	Description
	Cloud Connector over which the connection is opened. The default value is an empty string identifying the Cloud Connector that is connected without any location ID. This is also valid for all Cloud Connector versions prior to 2.9.0.
<code>URL.headers.<header-key></code>	<p>Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a <code>URL.headers</code> prefix separated by dot-delimiter. For example:</p> <div> <div>Sample Code</div> <pre>URL.headers.<header-key-1>=<header-value-1> ... URL.headers.<header-key-n>=<header-value-n></pre> </div> <div> <div>Note</div> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention. The HttpDestination (Version 2) [page 150] library supports this as of version 2.14.</p> </div>

Example

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

Related Information

[Principal Propagation \[page 130\]](#)

1.3.2.4 SAML Bearer Assertion Authentication

Context

SAP BTP supports applications to use the SAML Bearer assertion flow for consuming OAuth-protected resources. As a result, applications do not need to deal with some of the complexities of OAuth and can reuse existing identity providers for user data. Users are authenticated by using SAML against the configured trusted identity providers. The SAML assertion is then used to request an access token from an OAuth authorization server. This access token is automatically injected in all HTTP requests to the OAuth-protected resources.

→ Tip

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

Configuration Steps

You can create and configure an *OAuth2SAMLBearerAssertion* destination by using the properties listed below, and deploy it on SAP BTP. For more information, see:

- [Configure Destinations from the Cockpit \[page 76\]](#) or
- [Configure Destinations from the Console Client \[page 55\]](#)

ⓘ Note

Configuring *OAuth2SAMLBearerAssertion* destinations from the Eclipse IDE is not yet supported.

If you use the proxy type **OnPremise**, both OAuth server and the protected resource must be located on premise and exposed via the Cloud Connector. Make sure to set `URL` to the virtual address of the protected resource and `tokenServiceURL` to the virtual address of the OAuth server (see section *Properties* below).

ⓘ Note

The combination *on-premise* OAuth server and protected resource on the *Internet* is not supported, as well as OAuth server on the *Internet* and protected resource *on premise*.

Properties

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

Property	Description
Required	
Name	Destination name. It must be the same as the destination name you use for the configuration tools, that is, the console client and Destinations editor (cockpit).
Type	Destination type. Use HTTP as a value for all HTTP(S) destinations.
URL	URL of the protected resource on the called application
ProxyType	You can use the proxy types Internet or OnPremise . If you use OnPremise , the OAuth server must be accessed through the Cloud Connector [page 232].
Authentication	Authentication type. Use OAuth2SAMLBearerAssertion as a value.
audience	Intended audience for the assertion, which will be verified by the OAuth authorization server. For more information, see SAML 2.0 Bearer Assertion Profiles for OAuth 2.0 .
clientKey	Key that identifies the consumer to the authorization server
tokenServiceURL	URL of the OAuth server
tokenServiceUser	User for basic authentication to OAuth server (if required)
tokenServicePassword	Password for tokenServiceUser (if required)
Additional	
(Deprecated) SystemUser	User to be used when requesting access token from the OAuth authorization server. If this property is not specified, the currently logged-in user will be used.



⚠ Caution

This property is deprecated and will be removed soon. We recommend that you work on behalf of specific (named) users instead of working with a technical user.

As an alternative for technical user communication, we strongly recommend that you use one of these authentication types:

- Basic Authentication (see [Client Authentication Types for HTTP Destinations](#) [page 108])
- Client Certificate Authentication (see [Client Authentication Types for HTTP Destinations](#) [page 108])
- [OAuth Client Credentials Authentication](#) [page 110]

To extend an OAuth access token's validity, consider using an OAuth refresh token.

Property	Description
<code>nameQualifier</code>	Security domain of the user for which access token will be requested
<code>companyId</code>	Company identifier
<code>assertionIssuer</code>	Issuer of the SAML assertion
<code>authnContextClassRef</code>	Value of the <code>AuthnContextClassRef</code> tag, which is part of generated OAuth2SAMLBearerAssertion authentication. For more information, see SAML 2.0 specification  .
<code>nameIdFormat</code>	Value of the <code>NameIdFormat</code> tag, which is part of generated OAuth2SAMLBearerAssertion authentication. For more information, see SAML 2.0 specification  .
<code>userIdSource</code>	When this property is set, the generated SAML2 assertion uses the currently logged-in user as a value for the <code>NameId</code> tag.
<code>scope</code>	The value of the OAuth 2.0 <code>scope</code> parameter expressed as a list of space-delimited, case-sensitive strings.
<code>SkipSSOTokenGenerationWhenNoUser</code>	If this parameter is set and there is no user logged in, token generation is skipped, thus allowing anonymous access to public resources. If set, it may have any value.
<code>URL.headers.<header-key></code>	Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a <code>URL.headers</code> prefix separated by dot-delimiter. For example:

Sample Code

```
URL.headers.<header-key-1>=<header-value-1>
...
URL.headers.<header-key-n>=<header-value-n>
```

Note

This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention. The [HttpDestination \(Version 2\) \[page 150\]](#) library supports this as of version 2.14.

Note

When the OAuth authorization server is called, it accepts the trust settings of the destination. For more information, see [Server Certificate Authentication \[page 93\]](#).

Example

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

```
URL=https://demo.sapjam.com/OData/OData.svc
Name=sap_jam_odata
TrustAll=true
ProxyType=Internet
Type=HTTP
Authentication=OAuth2SAMLBearerAssertion
tokenServiceURL=https://demo.sapjam.com/api/v1/auth/token
clientKey=Aa1Bb2Cc3DdEe4F5GHIJ
audience=cubetree.com
nameQualifier=www.successfactors.com
```

Related Information

[Create HTTP Destinations \[page 79\]](#)

[Examples \(Cockpit\) \[page 89\]](#)

1.3.2.5 Application-to-Application SSO Authentication

Context

The *AppToAppSSO* destinations are used in scenario of application-to-application communication where the caller needs to propagate its logged-in user. Both applications are deployed on SAP BTP.

Configuration Steps

1. Configure your subaccount to allow principal propagation. For more information, see [Configure the Local Service Provider](#).

Note

This setting is done per subaccount, which means that once set to **Enabled** all applications within the subaccount will accept user propagation.

2. Create and configure an *AppToAppSSO* destination by using the properties listed below, and deploy it on SAP BTP. For more information, see:
 - [Configure Destinations from the Cockpit \[page 76\]](#)
 - [Configure Destinations from the Console Client \[page 55\]](#)

Note

Configuring *AppToAppSSO* destinations from the Eclipse IDE is not yet supported.

Properties

The following credentials need to be specified:

Property	Description
Name	Destination name. It must be the same as the destination name you use for the configuration tools, that is, the console client and <i>Destinations</i> editor (cockpit).
Type	Destination type. Use HTTP as a value for all HTTP(S) destinations.
Authentication	Authentication type. Use AppToAppSSO as a value.
URL	URL of the protected resource on the called application

Property	Description
SessionCookieNames	<p>Optional.</p> <p>The AppToApp authentication module will use it to recognize the user session which improves the performance of the HTTP client.</p> <div> <p>Note</p> <p>In case that a session cookie name has a variable part you can specify it as a regular expression.</p> </div> <p>You can specify more than one session cookie name as comma separated list:</p> <p><i>Example:</i></p> <pre>JSESSIONID, JTENANTSESSIONID.*, CookieName, Cookie*Name, CookieName.*</pre> <div> <p>Note</p> <p>The spaces after comma are optional.</p> </div> <p>If several cookies are listed, the session is recognized as soon as all of them are available in the response from the server.</p> <div> <p>Note</p> <p>Recommended value for the target Java app on SAP BTP is: JTENANTSESSIONID.*, and for the HANA XS app is: xsId.*.</p> </div>
saml2_audience	<p>Specifies a local SAML 2.0 provider name of the subaccount which consumes the target application.</p> <div> <p>Note</p> <p>If not specified, both applications must be consumed in the same subaccount.</p> </div>
SkipSSOTokenGenerationWhenNoUser	<p>Optional.</p> <p>If this parameter is set and there is no user logged in, token generation is skipped, thus allowing anonymous access to public resources. If set, it may have any value.</p>

Property	Description
<code>URL.headers.<header-key></code>	<p>Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a <code>URL.headers</code> prefix separated by dot-delimiter. For example:</p> <div> <div>❏ Sample Code</div> <pre>URL.headers.<header-key-1>=<header-value-1> ... URL.headers.<header-key-n>=<header-value-n></pre> </div> <div> <div>📌 Note</div> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention. The HttpDestination (Version 2) [page 150] library supports this as of version 2.14.</p> </div>

Example

```
#
#Wed Jan 13 12:25:47 UTC 2016
Name=apptOapp
URL=https://someurl.com
ProxyType=Internet
Type=HTTP
SessionCookieNames=JTENANTSESSIONID_.*
Authentication=AppToAppSSO
```

Related Information

[Server Certificate Authentication \[page 93\]](#)

[HTTP Proxy for On-Premise Connectivity \[page 153\]](#)

[AuthenticationHeaderProvider API \[page 142\]](#)

1.3.2.6 Client Authentication Types for HTTP Destinations

Context

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

No Authentication

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

`Authentication=NoAuthentication`

Note

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

Basic Authentication

This is used for destinations that refer to a service on the Internet or an on-premise system that requires basic authentication. The relevant property value is:

`Authentication=BasicAuthentication`

The following credentials need to be specified:

Caution

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

Property	Description
User	User name of the technical user to be used.
Password	Password of the technical user to be used.

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

📘 Note

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

📘 Note

Basic Authentication and **No Authentication** can be used in combination with `ProxyType=OnPremise`. In this case, also the `CloudConnectorLocationId` property can be specified. As of Cloud Connector 2.9.0, you can connect multiple Cloud Connectors to a subaccount as long as their location ID is different. The value defines the location ID identifying the Cloud Connector over which the connection shall be opened. The default value is the empty string identifying the Cloud Connector that is connected without any location ID. This is also the case for all Cloud Connector versions prior to 2.9.0.

Client Certificate Authentication

This is used for destinations that refer to a service on the Internet. The relevant property value is:

`Authentication=ClientCertificateAuthentication`

The following credentials need to be specified:

Property	Description
KeyStoreLocation 1. When used in local environment 2. When used in cloud environment	Path to the JKS file that contains the client certificate(s) for authentication against a remote server. 1. The relative path to the JKS file. The root path is the server's location on the file system. 2. The name of the JKS file.
KeyStorePassword	The password for the key storage. This property is mandatory in case <code>KeyStoreLocation</code> is used.

📘 Note

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

Configuration

- [Configure Destinations from the Cockpit \[page 76\]](#)
- [Configure Destinations from the Eclipse IDE \[page 63\]](#)
- [Configure Destinations from the Console Client \[page 55\]](#)

Related Information

[Server Certificate Authentication \[page 93\]](#)

1.3.2.7 OAuth Client Credentials Authentication

Create and configure an *OAuth2ClientCredentials* destination to consume OAuth-protected resources.

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

The client credentials are used to request an access token from an OAuth authorization server. If you use the [HttpDestination API and DestinationFactory \[page 137\]](#), the access token is automatically injected in all HTTP requests to the OAuth-protected resources. If you use the [ConnectivityConfiguration API \[page 140\]](#), you must retrieve the access token manually, using the [AuthenticationHeaderProvider API \[page 142\]](#) and inject it in the HTTP requests.

Note

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

Configuration Steps

You can create and configure an *OAuth2ClientCredentials* destination using the properties listed below, and deploy it on SAP BTP. To create and configure a destination, follow the steps described in:

- [Configure Destinations from the Cockpit \[page 76\]](#), or
- [Configure Destinations from the Console Client \[page 55\]](#)

Note

Configuring *OAuth2ClientCredentials* destinations from the Eclipse IDE is not yet supported.

Properties

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

Property	Description
<i>Required</i>	
Name	Destination name. It must be the same as the destination name you use in your preferred configuration tool, that is, the console client or the Destinations editor (cockpit).
Type	Destination type. Use HTTP as value for all HTTP(S) destinations.
URL	URL of the protected resource on the called application.
ProxyType	You can use the proxy types Internet or OnPremise . If you use OnPremise , the OAuth server must be accessed through the Cloud Connector [page 232] .
Authentication	Authentication type. Use OAuth2ClientCredentials as value.
clientId	Client ID used to retrieve the access token.
clientSecret	Client secret for the Client ID.
tokenServiceURL	URL of the OAuth server.
<i>Optional</i>	
tokenServiceUser	User for basic authentication to OAuth server (if required).
tokenServicePassword	Password for tokenServiceUser (if required).

Property	Description
<code>URL.headers.<header-key></code>	<p>Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a <code>URL.headers</code> prefix separated by dot-delimiter. For example:</p> <div> <p>Sample Code</p> <pre>URL.headers.<header-key-1>=<header-value-1> ... URL.headers.<header-key-n>=<header-value-n></pre> </div> <div> <p>Note</p> <p>This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention. The HttpDestination (Version 2) [page 150] library supports this as of version 2.14.</p> </div>

Note

When the OAuth authorization server is called, it accepts the trust settings of the destination, see [Server Certificate Authentication \[page 93\]](#).

Example

Sample Code

```
URL=https://demo.sapjam.com/odata/odata.svc
Name=sap_jam_odata
TrustAll=true
ProxyType=Internet
Type=HTTP
Authentication=OAuth2ClientCredentials
tokenServiceURL=http://demo.sapjam.com/api/v1/auth/token
tokenServiceUser=tokenServiceuser
tokenServicePassword=pass
clientId=clientId
clientSecret=secret
```

1.3.2.8 OAuth with X.509 Client Certificates

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

⚠ Caution

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

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

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

⚠ Caution

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

Supported Certificate Configuration Formats

- Java Keystore (.jks): Requires the `tokenService.KeyStorePassword` property.
- PKCS12 (.pfx or .p12): Requires the `tokenService.KeyStorePassword` property.

Supported OAuth Flows

- [SAML Bearer Assertion Authentication \[page 101\]](#)
- [OAuth Client Credentials Authentication \[page 110\]](#)

1.3.3 RFC Destinations

RFC destinations provide the configuration needed for communicating with an on-premise ABAP system via RFC. The RFC destination data is used by the JCo version that is offered within SAP BTP to establish and manage the connection.

RFC Destination Properties

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

- [User logon properties \[page 114\]](#)
- [Physical connection \[page 119\]](#)
- [Special parameters \[page 123\]](#)
- [Pooling configuration \[page 116\]](#)
- [Repository configuration \[page 118\]](#)
- [Principal Propagation SSO Authentication for RFC \[page 125\]](#)

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

Example

```
Name=SalesSystem
Type=RFC
jco.client.client=000
jco.client.lang=EN
jco.client.user=consultant
jco.client.passwd=<password>
jco.client.ashost=sales-system.cloud
jco.client.sysnr=42
jco.destination.pool_capacity=5
jco.destination.peak_limit=10
```

1.3.3.1 User Logon Properties

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

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

Property	Description
<code>jco.client.client</code>	Represents the client to be used in the ABAP system. Valid format is a three-digit number.
<code>jco.client.lang</code>	Optional property. Represents the logon language. If the property is not provided, the user's or system's default language is used. Valid values are two-character ISO language codes or one-character SAP language codes.
<code>jco.client.user</code>	Represents the user to be used for logging on to the ABAP system. Max. 12 characters long. <div> Note When working with the <i>Destinations</i> editor in the cockpit, enter the value in the <code><User></code> field. Do not enter it as additional property. </div>
<code>jco.client.alias_user</code>	Represents the user to be used for logging on to the ABAP system. Either <code>jco.client.user</code> or <code>jco.client.alias_user</code> must be specified. The alias user may be up to 40 characters long. <div> Note When working with the <i>Destinations</i> editor in the cockpit, enter the value in the <code><Alias User></code> field. Do not enter it as additional property. </div>
<code>jco.client.passwd</code>	Represents the password of the user that is used. <div> Note Passwords in systems of SAP NetWeaver releases lower than 7.0 are case-insensitive and can be only eight characters long. For releases 7.0 and higher, passwords are case-sensitive with a maximum length of 40. </div> <div> Note When working with the <i>Destinations</i> editor in the cockpit, enter this password in the <code><Password></code> field. Do not enter it as additional property. </div>

Property	Description
<code>jco.client.tls_client_certificate_logon</code>	<p>When set to 1, the client certificate provided by the <i>KeyStore</i>, which must be configured in addition, is used for authentication instead of <code>jco.client.user/jco.client.alias_user</code> and <code>jco.client.passwd</code>. This property is only relevant for a connection using WebSocket RFC (<code><Proxy Type>=Internet</code>).</p> <p>The default value is 0.</p> <div> <p>Note</p> <p>When working with the Destinations editor in the cockpit, the <code><User></code>, <code><Alias User></code> and <code><Password></code> fields are hidden when setting the property to 1.</p> </div> <p>For more information on WebSocket RFC, see also: WebSocket RFC</p>
<code>jco.destination.auth_type</code>	<p>Optional property.</p> <ul style="list-style-type: none"> If the property is not provided, its default value CONFIGURED_USER is used, which means that user, password, or other credentials are specified directly. To enable single sign-on via principal propagation (which means that the identity logged on in the cloud application is forwarded to the on-premise system), set the value to PrincipalPropagation. In this case, you do not need to provide <code>jco.client.user</code> and <code>jco.client.passwd</code> in the configuration. <div> <p>Note</p> <p>For PrincipalPropagation, you should configure the properties <code>jco.destination.repository.user</code> and <code>jco.destination.repository.passwd</code> instead, since there are special permissions needed (for metadata lookup in the back end) that not all business application users might have.</p> </div>

1.3.3.2 Pooling Configuration

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

Overview

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

Property	Description
<code>jco.destination.pool_capacity</code>	Represents the maximum number of idle connections kept open by the destination. A value of 0 has the effect of no connection pooling, that is, connections will be closed after each request. The default value is 1 .
<code>jco.destination.peak_limit</code>	<p>Represents the maximum number of active connections you can create for a destination simultaneously. Value 0 allows an unlimited number of active connections. Otherwise, if the value is less than the value of <code>jco.destination.pool_capacity</code>, it will be automatically increased to this value.</p> <p>Default setting is the value of <code>jco.destination.pool_capacity</code>. If <code>jco.destination.pool_capacity</code> is not specified, the default is 0 (unlimited).</p>
<code>jco.destination.max_get_client_time</code>	Represents the maximum time in milliseconds to wait for a free connection in case the maximum number of active connections is already allocated by applications. The default value is 30000 (30 seconds).
<code>jco.destination.expiration_time</code>	Represents the time in milliseconds after which idle connections that are available in the pool can be closed. The default value is 60000 (60 seconds).
<code>jco.destination.expiration_check_period</code>	Represents the interval in milliseconds for the timeout checker thread to check the idle connections in the pool for expiration. The default value is 60000 (60 seconds).
<code>jco.destination.pool_check_connection</code>	When setting this value to 1 , a pooled connection will be checked for corruption before being used for the next function module execution. Thus, it is possible to recognize corrupted connections and avoid exceptions being passed to applications when connectivity is basically working (default value is 0).

Note

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

Pooling Details

- Each destination is associated with a connection factory and, if the pooling feature is used, with a connection pool.
- Initially, the destination's connection pool is empty, and the JCo runtime does not preallocate any connection. The first connection will be created when the first function module invocation is performed. The `peak_limit` property describes how many connections can be created simultaneously, if applications allocate connections in different sessions at the same time. A connection is allocated either when a stateless function call is executed, or when a connection for a stateful call sequence is reserved within a session.
- After the `<peak_limit>` number of connections has been allocated (in `<peak_limit>` number of sessions), the next session will wait for at most `<max_get_client_time>` milliseconds until a different session releases a connection (either finishes a stateless call or ends a stateful call sequence). In case the waiting session does not get any connection during the `<max_get_client_time>` period, the function request will be aborted with *JCoException* with the key `JCO_ERROR_RESOURCE`.
- Connections that are no longer used by applications are returned to the destination pool. There is at most a `<pool_capacity>` number of connections kept open by the pool. Further connections (`<peak_limit>` - `<pool_capacity>`) will be closed immediately after usage. The pooled connections (open connections in the pool) are marked as expired if they are not used again during `<expiration_time>` milliseconds. All expired connections will be closed by a timeout checker thread which executes the check every `<expiration_check_period>` milliseconds.

1.3.3.3 Repository Configuration

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

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

Property	Description
<code>jco.destination.repository_destination</code>	Specifies which destination should be used for repository queries. If the destination does not exist, an error occurs when trying to retrieve the repository. Defaults to itself.

Property	Description
<code>jco.destination.repository.user</code>	<p>Optional property. If this property is set, and the repository destination is not set, it will be used as the user for repository queries. This configuration option allows using a different user for repository lookups with a single destination configuration, and restricting this user's permissions accordingly. See also SAP Note 460089.</p> <div> <p>Note</p> <p>When working with the <i>Destinations</i> editor in the cockpit, enter the value in the <code><Repository User></code> field. Do not enter it as additional property.</p> </div>
<code>jco.destination.repository.passwd</code>	<p>Represents the password for a repository user. If you use such a user, this property is mandatory.</p> <div> <p>Note</p> <p>When working with the <i>Destinations</i> editor in the cockpit, enter this password in the <code><Repository Password></code> field. Do not enter it as additional property.</p> </div>

1.3.3.4 Target System Configuration

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

Content

[Overview \[page 119\]](#)

[Proxy Types \[page 120\]](#)

[Direct Connection \[page 120\]](#)

[Load Balancing Connection \[page 121\]](#)

[WebSocket Connection \[page 122\]](#)

Overview

You can use the following configuration types alternatively:

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

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

Back to [Content \[page 119\]](#)

Proxy Types

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

Note

On SAP BTP trial (`hanatrial.ondemand.com`), you must add the parameter `jco.destination.proxy_type` with value `OnPremise` manually in section `<Additional Properties>`.

Back to [Content \[page 119\]](#)

Direct Connection

To use a direct connection to an application server over Cloud Connector, you must set the value for `<Proxy Type>` to `OnPremise`.

Property	Description
<code>jco.client.ashost</code>	Represents the application server host to be used. For configurations on SAP BTP, the property must match a virtual host entry in the Cloud Connector <i>Access Control</i> configuration. The property indicates that a direct connection is established.

Property	Description
<code>jco.client.sysnr</code>	Represents the so-called "system number" and has two digits. It identifies the logical port on which the application server is listening for incoming requests. For configurations on SAP BTP, the property must match a virtual port entry in the Cloud Connector Access Control configuration.

Note

The virtual port in the above access control entry must be named `sapgw<##>`, where `<##>` is the value of `sysnr`.

Back to [Content \[page 119\]](#)

Load Balancing Connection

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

Property	Description
<code>jco.client.mshost</code>	Represents the message server host to be used. For configurations on SAP BTP, the property must match a virtual host entry in the Cloud Connector Access Control configuration. The property indicates that load balancing is used for establishing a connection.
<code>jco.client.group</code>	Optional property. Identifies the group of application servers that is used, the so-called "logon group". If the property is not specified, the group <code>PUBLIC</code> is used.
<code>jco.client.r3name</code>	Represents the three-character system ID of the ABAP system to be addressed. For configurations on SAP BTP, the property must match a virtual port entry in the Cloud Connector Access Control configuration.

Note

The virtual port in the above access control entry must be named `sapms<###>`, where `<###>` is the value of `r3name`.

Property	Description
<code>jco.client.msserv</code>	Represents the port on which the message server is listening for incoming requests. you can use this property as an alternative to <code>jco.client.r3name</code> . One of these two must be present. For configurations on SAP BTP, the property must match a virtual port entry in the Cloud Connector Access Control configuration. You can therefore avoid look-ups in the <code>/etc/services</code> file (<code><Install_Drive>\Windows\System32\drivers\etc\services</code>) on the Cloud Connector host.

Back to [Content \[page 119\]](#)

WebSocket Connection

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

Prerequisites

- Your target system is an ABAP server as of S/4HANA (on-premise) version 1909.
- Your Java runtime version is at least one of the following:
 - **2.181.8** for *Java EE 6 Web Profile*
 - **2.146.17** for *Java Web Tomcat 7*
 - **3.101.18** for *Java Web Tomcat 8*
 - **1.70.18** for *Java EE 7 Web Profile TomEE 7*

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

Note

We recommend that you **do not use value 1 in productive scenarios**, but only for demo purposes.

Property	Description
TrustStoreLocation <ol style="list-style-type: none"> When used in local environment When used in cloud environment 	<p>If you don't want to use the standard JDK trust store as default (option <i>Use default JDK truststore</i> is unchecked), you must enter a <code><Trust Store Location></code>. This field indicates the path to the JKS file which contains trusted certificates (Certificate Authorities) for authentication against a remote client.</p> <ol style="list-style-type: none"> The relative path to the JKS file. The root path is the server's location on the file system. The name of the JKS file. <div> <p>Note</p> <p>The default JDK trust store is appended to the trust store defined in the destination configuration. As a result, the destination simultaneously uses both trust stores. If the <code><Trust Store Location></code> is not specified, the JDK trust store is used as default trust store for the destination.</p> </div>
TrustStorePassword	Password for the JKS trust store file. This property is mandatory if <code><Trust Store Location></code> is used.

Note

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

Note

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

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

Back to [Content \[page 119\]](#)

1.3.3.5 Parameters Influencing Communication Behavior

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

All properties are optional.

Property	Description
<code>jco.client.trace</code>	Defines whether protocol traces are created. Valid values are 1 (trace is on) and 0 (trace is off). The default value is 0 .
<code>jco.client.codepage</code>	Declares the 4-digit SAP codepage that is used when initiating the connection to the backend. The default value is 1100 (comparable to iso-8859-1). It is important to provide this property if the password that is used contains characters that cannot be represented in 1100 .
<code>jco.client.delta</code>	Enables or disables table parameter delta management. It is enabled if set to 1 , and respectively disabled if set to 0 . The default value is 1 .
<code>jco.client.cloud_connector_version</code>	Defines the Cloud Connector version used for establishing a connection to the on-premise system. The default value is 2 . Currently, no other values are supported.
<code>jco.client.cloud_connector_location_id</code>	As of Cloud Connector 2.9.0, you can connect multiple Cloud Connectors to a subaccount as long as their location ID is different. The location ID specifies the Cloud Connector over which the connection is opened. The default value is an empty string identifying the Cloud Connector that is connected without any location ID. This is also valid for all Cloud Connector versions prior to 2.9.0.
	<div> Note <p>When working with the Destinations editor in the cockpit, enter the Cloud Connector location ID in the <code><Location ID></code> field. Do not enter it as additional property.</p> </div>
<code>jco.client.serialization_format</code>	Defines the serialization format that is used when transferring function module data to the partner system. The property impacts the serialization behavior of function module data. Valid values are columnBased and rowBased . If you choose columnBased , the <i>fast RFC</i> serialization is used, as long as the partner system supports it, see SAP Note 2372888 . When choosing the rowBased option, <i>classic</i> or <i>basXML</i> serialization are used. The default value is rowBased .
<code>jco.client.network</code>	Defines which network type is expected to be used for the destination. The property impacts the serialization behavior of function module data, see SAP Note 2372888 . Valid values are WAN and LAN . The default value is LAN .

1.3.3.6 Principal Propagation SSO Authentication for RFC

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

Context

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

Note

This authentication type applies only for on-premise connectivity.

Configuration Steps

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

Properties

The following credentials need to be specified:

Property	Description
Name	Destination name. Must be unique for the destination level.
Type	Destination type. Use RFC for all RFC destinations.
jco.destination.auth_type	Authentication type. Use PrincipalPropagation as value.
jco.destination.proxy_type	You can only use proxy type OnPremise .


Example

```
Name=SalesSystem
Type=RFC
jco.client.client=000
jco.destination.auth_type=PrincipalPropagation
jco.destination.proxy_type=OnPremise
```

```
jco.client.ashost=sales-system.cloud
jco.client.sysnr=42
```



1.3.4 LDAP Destinations

For your cloud applications, you can use LDAP-based user management if you are operating an LDAP server within your network.

LDAP destinations carry connectivity details for accessing systems over *Lightweight Directory Access Protocol* (LDAP) as specified in [RFC 4511](#) . In combination with the Cloud Connector they enable SAP BTP applications to access LDAP servers in an on-premise corporate network. LDAP destinations are intended to be used with the Java *JNDI/LDAP Service Provider*.

For more information on how to use the Java JNDI/LDAP Service Provider see: <http://docs.oracle.com/javase/7/docs/technotes/guides/jndi/jndi-ldap.html> .

Tasks


Task Type	Task
 Operator and/or Developer	Using LDAP Destination Properties [page 126]
	Configuring Connectivity for LDAP [page 127]
 Developer	Consuming Connectivity for LDAP [page 128]

Using LDAP Destination Properties

The following properties are predefined for LDAP destinations:

Predefined LDAP Destination Properties

Name (UI)	Name (internal)	Description/Values	Corresponding Java JNDI/ LDAP service provider property
Authentication	ldap.authentication	Possible values: <i>BasicAuthentication</i> or <i>NoAuthentication</i>	<code>java.naming.security.authentication</code> with value simple in case of <i>BasicAuthentication</i> and value none in case of <i>NoAuthentication</i> .
Password	ldap.password		<code>java.naming.security.credentials</code>
Proxy Type	ldap.proxyType	Possible values: <i>Internet</i> or <i>OnPremise</i>	In case proxy type is <i>OnPremise</i> , the resulting property is <code>java.naming.ldap.factory.socket</code> with value com.sap.core.connectivity.api.ldap.LdapOnPremiseSocketFactory .
URL	ldap.url	URL to the LDAP server. Must be in form: <i>ldap://<host>:<port></i> . Example: <i>ldap://ldapservers.example-company.com:389</i>	<code>java.naming.provider.url</code>
User	ldap.user	Must be set in suitable form for the LDAP server. E.g. for Microsoft Active Directory the value should be in the following format: <i><user_name>@<domain_name></i> . Example: <i>serviceuser@examplecompany.com</i>	<code>java.naming.security.credentials</code>

As additional properties in an LDAP destination, you can specify the properties defined by the Java JNDI/LDAP Service Provider. For more details regarding these properties see *Environment Properties* at <http://docs.oracle.com/javase/7/docs/technotes/guides/jndi/jndi-ldap.html>  I.

Back to [Tasks \[page 126\]](#)

Configuring Connectivity for LDAP

Configure an LDAP Destination for SAP BTP Applications



[Configure Destinations from the Console Client \[page 55\]](#)

[Configure Destinations from the Eclipse IDE \[page 63\]](#)

[Configure Destinations from the Cockpit \[page 76\]](#)

Configure Connectivity between an LDAP System and a Cloud Application



Configure the Cloud Connector in your on-premise network for LDAP:

[Configuring the Cloud Connector for LDAP \[page 290\]](#)

Back to [Tasks \[page 126\]](#)

Consuming Connectivity for LDAP



Consume the LDAP tunnel in a Java application, see [Using LDAP \[page 203\]](#).

Back to [Tasks \[page 126\]](#)

1.3.5 Mail Destinations

A mail destination is used to specify the mail server settings for sending or fetching e-mail, such as the e-mail provider, e-mail account, and protocol configuration.

The name of the mail destination must match the name used for the mail session resource. You can configure a mail destination directly in a destination editor or in a mail destination properties file. The mail destination then needs to be made available in the cloud. If a mail destination is updated, an application restart is required so that the new configuration becomes effective.

ⓘ Note

SAP does not act as e-mail provider. To use this service, please cooperate with an external e-mail provider of your choice.

Mail Destination Properties

The following properties are used to configure the mail destination:

Property	Description	Mandatory
Name	The name of the destination. The mail session that is configured by this mail destination is available by injecting the mail session resource <code>mail / <Name></code> . The name of the mail session resource must match the destination name. The recommended name for a mail destination is <code>Session</code> .	Yes
Type	The type of destination. It must be <code>MAIL</code> for mail destinations.	Yes
<code>mail.*</code>	<code>javax.mail</code> properties for configuring the mail session. To send e-mails, you must specify at least <code>mail.transport.protocol</code> and <code>mail.smtp.host</code> . To retrieve e-mails, you must specify at least <code>mail.store.protocol</code> , <code>mail.<protocol>.host</code> , and for POP3 <code>mail.pop3.port</code> .	Depends on the mail protocol used.
<code>mail.password</code>	Password that is used for authentication. The user name for authentication is specified by <code>mail.user</code> (a standard <code>javax.mail</code> property).	Yes, if authentication is used (<code>mail.smtp.auth=true</code> and generally for fetching e-mail).

Note the following points:

- `mail.smtp.port`: The SMTP standard ports 465 (SMTPS) and 587 (SMTP+STARTTLS) are open for outgoing connections on SAP BTP.
- `mail.pop3.port`: The POP3 standard ports 995 (POP3S) and 110 (POP3+STARTTLS) are open for outgoing connections (used to fetch e-mail).
- `mail.imap.port`: The IMAP standard ports 993 (IMAPS) and 143 (IMAP +STARTTLS) are open for outgoing connections (used to fetch e-mail).
- `mail.<protocol>.host`: The mail server of an e-mail provider accessible on the Internet, such as Google Mail (for example, `smtp.gmail.com`, `imap.gmail.com`, and so on).

SMTP and IMAP Example

The destination below has been configured to use Gmail as the e-mail provider, SMTP with STARTTLS (port 587) for sending e-mail, and IMAP (SSL) for receiving e-mail:

```
Name=Session
Type=MAIL
mail.user=<gmail account name>
mail.password=<gmail account password>
mail.transport.protocol=smtp
mail.smtp.host=smtp.gmail.com
```

```
mail.smtp.auth=true
mail.smtp.starttls.enable=true
mail.smtp.port=587
mail.store.protocol=imaps
mail.imaps.host=imap.gmail.com
```

SMTPS Example

The destination below uses Gmail and SMTPS (port 465) for sending e-mail:

```
Name=Session
Type=MAIL
mail.user=<gmail account name>
mail.password=<gmail account password>
mail.transport.protocol=smtps
mail.smtps.host=smtp.gmail.com
mail.smtps.auth=true
mail.smtps.port=465
```

Related Information

[JavaMail API Documentation](#) ➔

[Configure Destinations from the Eclipse IDE \[page 63\]](#)

[Configure Destinations from the Cockpit \[page 76\]](#)

[Configure Destinations from the Console Client \[page 55\]](#)

1.3.6 Principal Propagation

Forward the identity of cloud users to an on-premise system to enable single sign-on (Neo environment).

Content

[Overview \[page 131\]](#)

[How It Works \[page 131\]](#)

[Using the Principal Propagation Property in Destinations \[page 132\]](#)

[Tasks \[page 132\]](#)

Overview

The Connectivity service provides a secure way of forwarding the identity of a cloud user to the Cloud Connector, and from there to an on-premise system. This process is called **principal propagation**.

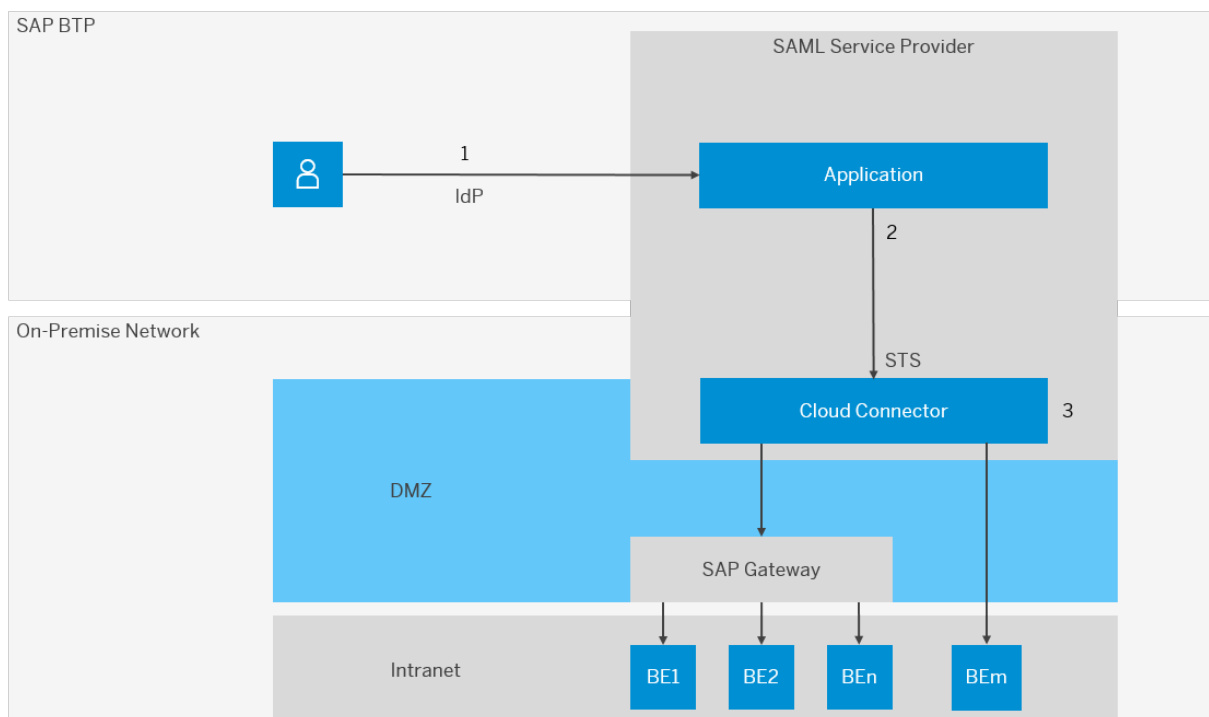
It uses a SAML token as exchange format for the user information. User mapping is done in the back end. The token is forwarded either directly, or an x.509 certificate is generated, which is then used in the backend.

⚠ Restriction

This authentication is only applicable if you connect to your on-premise system via the Cloud Connector.

Back to [Content \[page 130\]](#)

How It Works



1. The user authenticates at the cloud application front end via the IdP (Identity Provider) using a standard SAML Web SSO profile. When the backend connection is established by the cloud application, the destination service (re)uses the received SAML assertion to create the connection to the on-premise backend system (BE1-BE_m).
2. The Cloud Connector validates the received SAML assertion for a second time, extracts the attributes, and uses its STS (Security Token Service) component to issue a new token (an x.509 certificate) with the same or similar attributes to assert the identity to the backend.
3. The Cloud Connector and the cloud application share the same SAML service provider identity, which means that the trust is only set up once in the IdP.

Back to [Content \[page 130\]](#)

Using the Principal Propagation Property in Destinations

You can create and configure connectivity destinations using the `PrincipalPropagation` property in the Eclipse IDE and in the cockpit. Keep in mind that this property is only available for destination configurations created in the cloud.

- [Create and Delete Destinations on the Cloud \[page 68\]](#) (Eclipse IDE, procedure and examples)
- [Create Destinations \(Cockpit\) \[page 79\]](#) (procedure and examples)

Back to [Content \[page 130\]](#)

Tasks

- [Configuring Principal Propagation \[page 304\]](#) (Cloud Connector)
- [Principal Propagation SSO Authentication for HTTP \[page 99\]](#) (SAP BTP destination)
- [Principal Propagation SSO Authentication for RFC \[page 125\]](#) (SAP BTP destination)

Back to [Content \[page 130\]](#)

Related Information

[Authentication Configuration](#)

1.3.7 Multitenancy in the Connectivity Service

Using multitenancy for applications that require a connection to a remote service or on-premise application.

Endpoint Configuration

Applications that require a connection to a remote service can use the Connectivity service to configure HTTP or RFC endpoints. In a provider-managed application, such an endpoint can either be once defined by the

application provider ([Provider-Specific Destination \[page 133\]](#)), or by each application consumer ([Consumer-Specific Destination \[page 134\]](#)).

If the application needs to use the same endpoint, independently from the current application consumer, the destination that contains the endpoint configuration is uploaded by the application provider. If the endpoint should be different for each application consumer, the destination can be uploaded by each particular application consumer.

To prevent application consumers from using an individual endpoint for a provider application, you can set the property `DestinationProvider=Application` in the HTTP or RFC destination. In this case, the destination is always read from the provider application.

Note

This connectivity type is fully applicable also for on-demand to on-premise connectivity.

Destination Levels

You can configure destinations simultaneously on three levels: *subscription*, *consumer subaccount* and *application*. This means that it is possible to have one and the same destination on more than one configuration level. For more information, see [Managing Destinations \[page 54\]](#).

Destination visibility according to the level, when uploaded on:

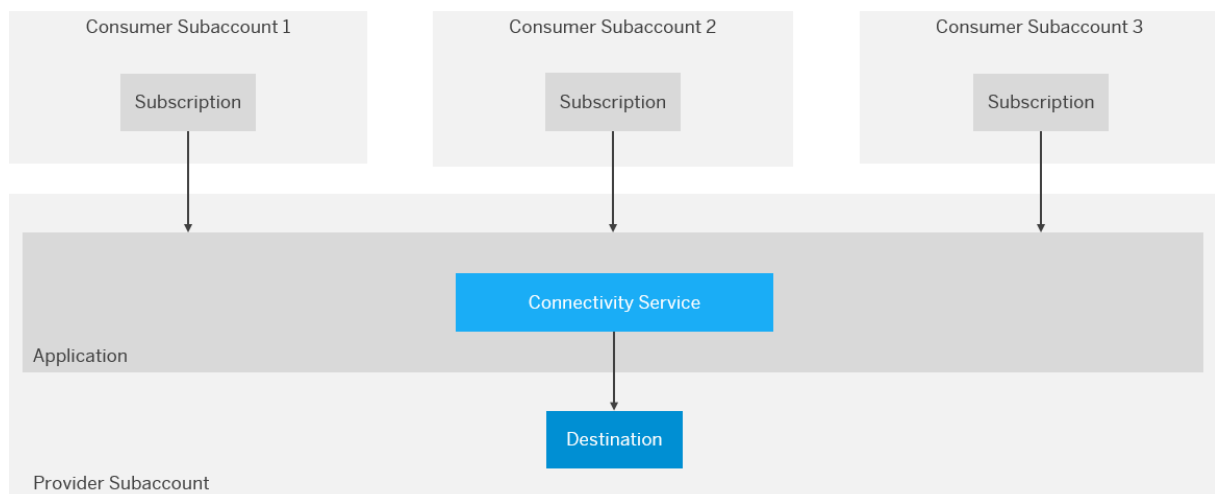
Level	Visibility
Subscription level	Only visible for the dedicated subscription.
Subaccount level	Visible for the whole consumer subaccount.
Application level	Visible by all tenants and subaccounts, regardless their permission settings.

When the application accesses the destination at runtime, the Connectivity service

1. looks up the requested destination in the consumer subaccount on *subscription* level. If no destination is available there, it
2. checks if the destination is available on the *subaccount* level of the consumer subaccount. If there is still no destination found, it
3. searches on *application* level of the provider subaccount.

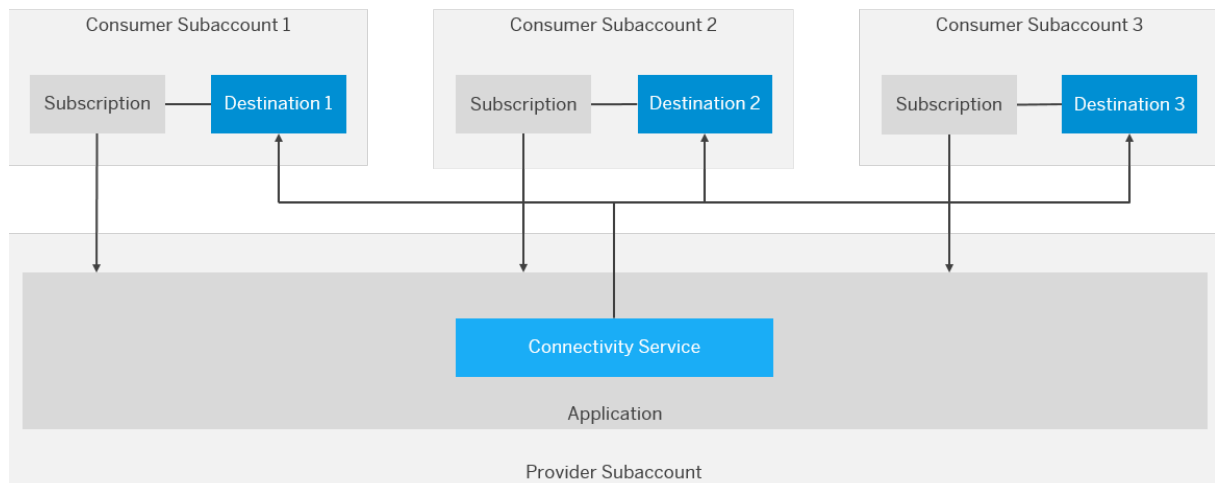
Back to [Top \[page 132\]](#)

Provider-Specific Destination



Back to [Top \[page 132\]](#)

Consumer-Specific Destination



Back to [Top \[page 132\]](#)

Related Information

[Developing Multitenant Applications in the Neo Environment](#)
[Create a Multitenant Connectivity Application](#)

1.3.8 Configuring Backup

Find an overview of backup procedures for your destination configurations.

Export Destinations (Cockpit) [page 88]	Export destinations from the cockpit.
Export Destinations (IDE) [page 72]	Export destinations from the IDE.
Download Destinations [page 59]	Download destinations from the console client.

1.4 Development

Consume the Connectivity service from a Java or HANA XS application in the Neo environment and use the Destination Configuration service to provide the required destination information.

Task	Description
Consuming the Connectivity Service (Java) [page 135]	Connect your Java cloud applications to the Internet, make cloud-to-on-premise connections to SAP or non-SAP systems, or send and fetch e-mail.
Consuming the Connectivity Service (HANA XS) [page 217]	Create connectivity destinations for HANA XS applications, configure security, add roles and test them in an enterprise or trial landscape.
Consuming the Destination Configuration Service [page 231]	Retrieve destination configurations for your cloud application in the Neo environment, in a secure and reliable way.

1.4.1 Consuming the Connectivity Service (Java)

Connect your Java cloud applications to the Internet, make cloud-to-on-premise connections to SAP or non-SAP systems, or send and fetch e-mail.

Task	Description
Connectivity and Destination APIs [page 136]	Find an overview of the available connectivity and destination APIs.
Exchanging Data via HTTP [page 147]	Consume the Connectivity service using the HTTP protocol.
Invoking ABAP Function Modules via RFC [page 191]	Call a remote-enabled function module in an ABAP server using the SAP Java Connector (JCo) API.
Using LDAP [page 203]	You can use LDAP-based user management if you are operating an LDAP server within your local network.

Task	Description
Using the TCP Protocol for Cloud Applications [page 205]	Access on-premise systems via TCP-based protocols, using a SOCKS5 proxy.
Sending and Fetching E-Mail [page 208]	Send mail messages from your cloud applications using e-mail providers that are accessible on the Internet.

1.4.1.1 Connectivity and Destination APIs

Destinations

Destinations are part of SAP Connectivity service and are used for the outbound communication from a cloud application to a remote system. They contain the connection details for the remote communication of an application, which can be configured for each customer to accommodate the specific customer back-end systems and authentication requirements. For more information, see [Managing Destinations \[page 54\]](#).

Destinations should be used by application developers when they aim to provide applications that:

- Integrate with remote services or back-end systems that need to be configured by customers
- Integrate with remote services or back-end systems that are located in a fenced environment (that is, behind firewalls and not publicly accessible)

→ Tip

HTTP clients created by destination APIs allow parallel usage of HTTP client instances (via class `ThreadSafeClientConnManager`).

Connectivity APIs

Package	Description
<code>org.apache.http</code>	http://hc.apache.org ➡
<code>org.apache.http.client</code>	http://hc.apache.org/httpcomponents-client-ga/httpclient/apidocs/org/apache/http/client/package-summary.html ➡
<code>org.apache.http.util</code>	http://hc.apache.org/httpcomponents-core-ga/httpcore/apidocs/org/apache/http/util/package-summary.html ➡
<code>javax.mail</code>	https://javamail.java.net/nonav/docs/api/ ➡

Package	Description
	The SAP BTP SDK for Java Web uses version 1.4.1 of <code>javax.mail</code> , the SDK for Java EE 6 Web Profile uses version 1.4.5 of <code>javax.mail</code> , and the SDK for Java Web Tomcat 7 uses version 1.4.7 of <code>javax.mail</code> .
<code>com.sap.core.connectivity.api</code>	You can find the Connectivity service API in directory <code><SDK_location>/javadoc/com/sap/core/connectivity/api</code> . For more information, see API Documentation .

Destination APIs

- [JavaMail API \[page 209\]](#)
- [HttpDestination Library \[page 148\]](#)
- [Java Connector API \[page 142\]](#)
- [ConnectivityConfiguration API \[page 140\]](#)
- [AuthenticationHeaderProvider API \[page 142\]](#)
- [Principal Propagation Using HTTP Proxy \[page 155\]](#)
- [HttpDestination API and DestinationFactory \[page 137\]](#)

Destination Configuration Tasks

- [Configure Destinations from the Eclipse IDE \[page 63\]](#)
- [Configure Destinations from the Console Client \[page 55\]](#)
- [Configure Destinations from the Cockpit \[page 76\]](#)
- [Consuming the Destination Configuration Service \[page 231\]](#)

1.4.1.1.1 HttpDestination API and DestinationFactory

All connectivity API packages are visible by default from all Web applications. Applications can consume the destinations via a JNDI lookup.

Procedure

Prerequisites

You have set up your Java development environment. See also: [Setting Up the Development Environment](#)

Retrieving HTTP Destinations Using HttpDestination API

To consume destinations using `HttpDestination` API, you need to define your destination as a resource in the `web.xml` file.

1. An example of a destination resource named **myBackend**, which is described in the `web.xml` file, is as follows:

```
<resource-ref>
  <res-ref-name>myBackend</res-ref-name>
  <res-type>com.sap.core.connectivity.api.http.HttpDestination</res-type>
</resource-ref>
```

2. In your servlet code, you can look up the destination (an HTTP destination in this example) from the JNDI registry as following:

```
import javax.naming.Context;
import javax.naming.InitialContext;
import com.sap.core.connectivity.api.http.HttpDestination;
...

// coding to lookup the destination "myBackend"
Context ctx = new InitialContext();
HttpDestination destination = (HttpDestination) ctx.lookup("java:comp/env/
myBackend");
```

Note

If you want the lookup name to differ from the destination name, you can specify the lookup name in `<res-ref-name>` and the destination name in `<mapped-name>`, as shown in the following example:

```
<resource-ref>
  <res-ref-name>myLookupName</res-ref-name>
  <res-type>com.sap.core.connectivity.api.http.HttpDestination</res-type>
  <mapped-name>myBackend</mapped-name>
</resource-ref>
```

3. With the retrieved HTTP destination, you can then, for example, send a simple GET request to the configured remote system by using the following code:

```
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.HttpResponse;
...

// coding to call service "myService" on the system configured in the given
destination
HttpClient createHttpClient = destination.createHttpClient();
HttpGet get = new HttpGet("myService");
HttpResponse resp = createHttpClient.execute(get);
```

Note

If you want to use `<res-ref-name>`, which contains `/`, the name after the last `/` should be the same as the destination name. For example, you can use `<res-ref-name>connectivity/myBackend</res-ref-name>`. In this case, you should use `java:comp/env/connectivity/myBackend` as a lookup string.

If you want to get the URL of your configured destination, use the `URI getURI()` method. This method returns the URL, defined in the destination configuration, converted to URI.

Retrieving HTTP Destinations Using DestinationFactory

As alternative approach how to retrieve an HTTP destination, `DestinationFactory` can be used. We recommend this approach if the used destinations are unknown at implementation time and should be loaded dynamically at runtime.

1. Define the `DestinationFactory` as a JNDI resource in the `web.xml` file:

```
<resource-ref>
  <res-ref-name>connectivity/DestinationFactory</res-ref-name>
  <res-type>com.sap.core.connectivity.api.DestinationFactory</res-type>
</resource-ref>
```

2. In your Java code, you can then look it up and use it in following way:

```
DestinationFactory destinationFactory = (DestinationFactory)
ctx.lookup(DestinationFactory.JNDI_NAME);
destination = (HttpDestination)
destinationFactory.getDestination(destinationName);
```

Note

If you have two destinations with the same name, one configured on subaccount level and the other on application level, the `getConfiguration()` method will return the destination on subaccount level.

The preference order is: subscription level -> subaccount level -> application level.

Related Information

If you need to also add Maven dependencies, take a look at this blog:

[Building Java Web Applications with Maven](#) 

See also:

[Maven Plugin](#)

1.4.1.1.2 ConnectivityConfiguration API

All connectivity API packages are visible by default from all Web applications. Applications can consume the connectivity configuration via a JNDI lookup.

Context

Besides making destination configurations, you can also allow your applications to use their own HTTP clients. The ConnectivityConfiguration API provides you a direct access to the destination configurations of your applications. This API also:

- Can be used independent of the existing destination API so that applications can bring and use their own HTTP client
- Consists of both a **public REST API** and a **Java client API**.

The ConnectivityConfiguration API is supported by all runtimes, including **Java Web Tomcat 7**. For more information about runtimes, see [Application Runtime Container](#).

To learn how to retrieve destination configurations, follow the procedure below.

Procedure

1. To consume connectivity configuration using JNDI, you need to define ConnectivityConfiguration API as a resource in the web.xml file. An example of a ConnectivityConfiguration resource named **connectivityConfiguration**, which is described in the web.xml file, is as follows:

```
<resource-ref>
  <res-ref-name>connectivityConfiguration</res-ref-name>
  <res-
type>com.sap.core.connectivity.api.configuration.ConnectivityConfiguration</
res-type>
</resource-ref>
```

2. In your servlet code, you can look up the ConnectivityConfiguration API from the JNDI registry as following:

```
import javax.naming.Context;
import javax.naming.InitialContext;
import com.sap.core.connectivity.api.configuration.ConnectivityConfiguration;
...

// look up the connectivity configuration API "connectivityConfiguration"
Context ctx = new InitialContext();
ConnectivityConfiguration configuration = (ConnectivityConfiguration)
ctx.lookup("java:comp/env/connectivityConfiguration");
```

3. With the retrieved ConnectivityConfiguration API, you can read all properties of any destination defined on subscription, application or subaccount level.

Note

If you have two destinations with the same name, one configured on subaccount level and the other on application level, the `getConfiguration()` method will return the destination on subaccount level. The preference order is: subscription level -> subaccount level -> application level.

```
// get destination configuration for "myDestinationName"
DestinationConfiguration destConfiguration =
configuration.getConfiguration("myDestinationName");
// get the "myDestinationName" authentication property (example)
String value = destConfiguration.getProperty("Authentication");
// get all destination properties
Map<String, String> allDestinationProperties =
destConfiguration.getAllProperties();
```

4. If truststore and keystore are defined in the corresponding destination, they can be accessed by using methods `getKeyStore` and `getTrustStore`.

```
// get destination configuration for "myDestinationName"
DestinationConfiguration destConfiguration =
configuration.getConfiguration("myDestinationName");

// get the configured keystore
KeyStore keyStore = destConfiguration.getKeyStore();

// get the configured truststore
KeyStore trustStore = destConfiguration.getTrustStore();

// create sslcontext
TrustManagerFactory tmf =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
tmf.init(trustStore);

KeyManagerFactory keyManagerFactory =
KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
// get key store password from destination
String keyStorePassword = destConfiguration.getProperty("KeyStorePassword");
keyManagerFactory.init(keyStore, keyStorePassword.toCharArray());

SSLContext sslContext = SSLContext.getInstance("TLS");
sslContext.init(keyManagerFactory.getKeyManagers(), tmf.getTrustManagers(),
null);
SSLSocketFactory sslSocketFactory = sslContext.getSocketFactory();

// get the destination URL
String value = destConfiguration.getProperty("URL");
URL url = new URL(value);

// use the sslcontext for url connection
URLConnection urlConnection = url.openConnection();
((HttpsURLConnection) urlConnection).setSSLSocketFactory(sslSocketFactory);
urlConnection.connect();
InputStream in = urlConnection.getInputStream();
...
```

1.4.1.1.3 Java Connector API

The Java Connector (JCo) is a middleware component that enables you to develop ABAP-compliant components and applications in Java.

JCo supports communication with Application Server ABAP (AS ABAP) in both directions:

- Inbound - Java calls ABAP
- Outbound - ABAP calls Java

The JCo can be implemented with desktop applications and Web server applications.

📘 Note

You can find generic information regarding authorizations required for the use of JCo in SAP Note [460089](#).

To learn in detail about the JCo API, see the JCo 3.1 documentation on [SAP Support Portal](#).

📘 Note

This documentation contains sections not applicable to SAP BTP. In particular:

- **Architecture:** CPIC is only used in the last mile from your Cloud Connector to the backend. From the cloud to the Cloud Connector, TLS-protected communication is used.
- **Installation:** SAP BTP already includes all the necessary artifacts.
- **Customizing and Integration:** In SAP BTP, the integration is already done by the runtime. You can concentrate on your business application logic.
- **Server Programming:** The programming model of JCo in SAP BTP does not include server-side RFC communication.
- **IDoc Support for External Java Applications:** For the time being, there is no IDocLibrary for JCo available in SAP BTP.

Related Information

[Invoking ABAP Function Modules via RFC \[page 191\]](#)

1.4.1.1.4 AuthenticationHeaderProvider API

Implement authentication token generation for your Web application using the `AuthenticationHeaderProvider` API.

Context

The `AuthenticationHeaderProvider` API allows your Web applications to use their own HTTP clients, providing authentication token generation for application-to-application SSO (single sign-on) and on-premise SSO.

This API:

- Provides additional helper methods, which facilitate the task to initialize an HTTP client (for example, an authentication method that helps you set headers for application-to-application SSO).
- Consists of both a public REST API and a Java client API. See also [API Documentation](#).

All connectivity API packages are visible by default from all Web applications. Applications can consume the authentication header provider via a JNDI lookup.

Note

The `AuthenticationHeaderProvider` API is supported by all runtimes, including **Java Web Tomcat 7**. For more information about runtimes, see [Application Runtime Container](#).

Tasks

[Retrieve Authentication Header Providers \[page 143\]](#)

[Generate Application-to-Application SSO Authentication \[page 144\]](#)

[Generate On-Premise SSO Authentication \[page 145\]](#)

[Generate SAPAssertionSSO Headers \[page 145\]](#)

[Generate OAuth2SAMLBearerAssertion Headers \[page 145\]](#)

[Generate Client Credentials Headers \[page 146\]](#)

Back to [Context \[page 143\]](#)

Retrieve Authentication Header Providers

1. To consume the `AuthenticationHeaderProvider` API using JNDI, you need to define it as a resource in the `web.xml` file. An example of an `AuthenticationHeaderProvider` resource named **myAuthHeaderProvider**, which is described in the `web.xml` file, looks like this:

```
<resource-ref>
  <res-ref-name>myAuthHeaderProvider</res-ref-name>
  <res-
type>com.sap.core.connectivity.api.authentication.AuthenticationHeaderProvider
</res-type>
```

```
</resource-ref>
```

2. In your servlet code, you can look up the `AuthenticationHeaderProvider` API from the JNDI registry:

```
import javax.naming.Context;
import javax.naming.InitialContext;
import
com.sap.core.connectivity.api.authentication.AuthenticationHeaderProvider;
...

// look up the connectivity authentication header provider resource called
"myAuthHeaderProvider"
Context ctx = new InitialContext();
AuthenticationHeaderProvider authHeaderProvider
= (AuthenticationHeaderProvider) ctx.lookup("java:comp/env/
myAuthHeaderProvider");
```

Back to [Tasks \[page 143\]](#)

Back to [Context \[page 143\]](#)

Generate Application-to-Application SSO Authentication

The `AuthenticationHeaderProvider` API can generate an authorization header to be used in a scenario of application-to-application communication where the caller needs to propagate its logged-in user. Both applications are deployed on SAP BTP and consumed within a single subaccount. The header must be embedded in the request to the target application.

→ Tip

We recommend that you pack the HTTP client (Apache or other) inside the `lib` folder of your Web application archive.

Prerequisites:

- Principal propagation must be enabled for the subaccount. For more information, see [Application Identity Provider](#) → section *Specifying Custom Local Provider Settings*.
- Both applications must run on behalf of the same subaccount.
- The receiving application must use SAML2 authentication.

📌 Note

If you work with the **Java Web Tomcat 7** runtime, bear in mind that the following code snippet works properly only when using the Apache HTTP client version **4.1.3**. If you use other (higher) versions of the Apache HTTP client, you should adapt your code.

```
// retrieve the authorization header for application-to-application SSO
AuthenticationHeader appToAppSSOHeader =
authHeaderProvider.getAppToAppSSOHeader(url);

// create an HTTP client and add the header to the request
HttpClient httpClient = new DefaultHttpClient();
```

```
HttpGet request = new HttpGet(url);
request.addHeader(appToAppSSOHeader.getName(), appToAppSSOHeader.getValue());

// execute the request
HttpResponse response = httpClient.execute(request);
```

Back to [Tasks \[page 143\]](#)

Back to [Context \[page 143\]](#)

Generate On-Premise SSO Authentication

To learn how to generate on-premise SSO authentication, see [Principal Propagation Using HTTP Proxy \[page 155\]](#).

Back to [Tasks \[page 143\]](#)

Back to [Context \[page 143\]](#)

Generate SAPAssertionSSO Headers

The aim of the SAPAssertionSSO headers is to generate an assertion ticket that propagates the currently logged-in SAP BTP user to an SAP back-end system. You can use this authentication type only if the user IDs on both sides are the same.

The `AuthenticationHeaderProvider` API provides the following method for generating such headers:

```
AuthenticationHeader getSAPAssertionHeader(DestinationConfiguration
destinationConfiguration);
```

See also: [SAP Assertion SSO Authentication \[page 96\]](#).

Back to [Tasks \[page 143\]](#)

Back to [Context \[page 143\]](#)

Generate OAuth2SAMLBearerAssertion Headers

SAP BTP supports applications to use the SAML Bearer assertion flow for consuming OAuth-protected resources. As a result, applications do not need to deal with some of the complexities of OAuth and can reuse existing identity providers for user data. Users are authenticated by using SAML against the configured trusted identity providers. The SAML assertion is then used to request an access token from an OAuth authorization server. This access token should be injected in all HTTP requests to the OAuth-protected resources.

Note

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

The `AuthenticationHeaderProvider` API provides the following method for generating such headers:

```
List<AuthenticationHeader>  
getOAuth2SAMLBearerAssertionHeaders(DestinationConfiguration  
destinationConfiguration);
```

See also: [SAML Bearer Assertion Authentication \[page 101\]](#).

Back to [Tasks \[page 143\]](#)

Back to [Context \[page 143\]](#)

Generate Client Credentials Headers

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

You can use the client credentials to request an access token from an OAuth authorization server. If you use the [HttpDestination API and DestinationFactory \[page 137\]](#), the access token is automatically injected in all HTTP requests to the OAuth-protected resources. If you use the [ConnectivityConfiguration API \[page 140\]](#), you must retrieve the access token using the `AuthenticationHeaderProvider` API and manually inject it in the HTTP requests.

Note

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

The `AuthenticationHeaderProvider` API provides the following method for generating such headers:

```
AuthenticationHeader getOAuth2ClientCredentialsHeader (DestinationConfiguration  
destinationConfiguration);
```

See also: [OAuth Client Credentials Authentication \[page 110\]](#).

Back to [Tasks \[page 143\]](#)

Back to [Context \[page 143\]](#)

Related Information

[HTTP Proxy for On-Premise Connectivity \[page 153\]](#)

1.4.1.2 Exchanging Data via HTTP

Consuming Connectivity via HTTP

- Call an Internet service using a simple application that queries some information from a public service:
[Consume Internet Services \(Java Web or Java EE 6 Web Profile\) \[page 156\]](#)
[Consume Internet Services \(Java Web Tomcat 7\) \[page 164\]](#)
- Call a service from a fenced customer network using a simple application that consumes an on-premise ping service:
[Consume Backend Systems \(Java Web or Java EE 6 Web Profile\) \[page 171\]](#)
[Consume Backend Systems \(Java Web Tomcat 7\) \[page 181\]](#)

Configuring Connectivity via HTTP

- Configure an application using destinations:
[Configure Destinations from the Eclipse IDE \[page 63\]](#)
[Configure Destinations from the Console Client \[page 55\]](#)
[Configure Destinations from the Cockpit \[page 76\]](#)
- Configure connectivity between a customer system and an on-demand application. You must install the Cloud Connector in your internal network and then configure it to expose an on-premise service. For more information, see [Initial Configuration \(HTTP\) \[page 285\]](#) and [Configure Access Control \(HTTP\) \[page 342\]](#).

Connecting to On-Premise Back-End Services

You can consume on-premise back-end services in two ways – via HTTP destinations and via the HTTP Proxy. For more information, see:

- [HTTP Destinations \[page 91\]](#)
- [HTTP Proxy for On-Premise Connectivity \[page 153\]](#)

Connecting to a Local Host

To create a loopback connection, you can use the dedicated HTTP port bound to localhost. The port number can be obtained from the cloud environment variable `HC_LOCAL_HTTP_PORT`.

For more information, see [Using Cloud Environment Variables](#) → section "List of Environment Variables".

Note

Note that when deploying locally from the Eclipse IDE or the console client, the HTTP port may differ.

Related Information

[Using the Keystore Service for Client Side HTTPS Connections](#)

1.4.1.2.1 HttpDestination Library

Learn about the different `HttpDestination` library versions available for the Connectivity service.


`HttpDestination` is a Java library that simplifies the consumption of destination configurations and provides smooth integration with on-premise systems for SAP BTP services like Security and Connectivity.

There are two major versions of the library:

- [HttpDestination \(Version 1\) \[page 148\]](#): Available in the Neo SDK, package `<SDK_location>/javadoc/com/sap/core/connectivity/api`.

Caution

`HttpDestination v1` is a deprecated legacy version. We recommend that you use `HttpDestination v2`.

- [HttpDestination \(Version 2\) \[page 150\]](#): based on Apache HttpClient 4.5, available in the [Maven Central Repository](#) .

Related Information

[Connectivity and Destination APIs \[page 136\]](#)

1.4.1.2.1.1 HttpDestination (Version 1)

Learn about the legacy `HttpDestination` library version available for the Connectivity service.

Overview

By default, all Connectivity API packages are visible from all Web applications. In this classical case, applications can consume the destinations via a JNDI lookup. For more information, see [Connectivity and Destination APIs \[page 136\]](#).

There are specific cases though, when the destination names are not known in advance and cannot be defined in the `web.xml` file. This is relevant to HTTP destinations, and in these cases, you must use the `DestinationFactory` JNDI lookup (`com.sap.core.connectivity.api.DestinationFactory`). To do this, follow the procedure below.

⚠ Caution

- If you use the SDK for Java Web, we only *recommend* that you create a destination before deploying the application.
- If you use the SDK for Java EE 6 Web Profile, you *must* create a destination before deploying the application.
- If you use the SDK for Java Web Tomcat 7, the `DestinationFactory` API is not supported. Instead, you can use the [ConnectivityConfiguration API \[page 140\]](#).
- If you use the SDK for Java Web Tomcat 8, you can use the [ConnectivityConfiguration API \[page 140\]](#). We recommend that you use [HttpDestination \(Version 2\) \[page 150\]](#).

→ Tip

If you know in advance the names of all destinations you need, you should rather use destinations. Otherwise, we recommend that you use `DestinationFactory`.

Procedure

To look up the destination factory using JNDI, follow these steps:

1. For your application, define a reference in the `web.xml` file:

📄 Sample Code

```
<resource-ref>
  <res-ref-name>connectivity/DestinationFactory</res-ref-name>
  <res-type>com.sap.core.connectivity.api.DestinationFactory</res-type>
</resource-ref>
```

2. Use the following code to look up the `DestinationFactory`:

📄 Sample Code

```
import com.sap.core.connectivity.api.DestinationFactory;
import com.sap.core.connectivity.api.http.HttpDestination
```

```
...
Context ctx = new InitialContext();
DestinationFactory destinationFactory
=(DestinationFactory)ctx.lookup(DestinationFactory.JNDI_NAME);
HttpDestination destination = (HttpDestination)
destinationFactory.getDestination("myBackend");
```

3. With the retrieved HTTP destination, you can then, for example, send a simple [GET](#) request to the configured remote system by using the following code:

Sample Code

```
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.HttpResponse;
...

// coding to call service "myService" on the system configured in the
given destination
HttpClient createHttpClient = destination.createHttpClient();
HttpGet get = new HttpGet("myService");
HttpResponse resp = createHttpClient.execute(get);
```

1.4.1.2.1.2 HttpDestination (Version 2)

Use the current `HttpDestination` library version available for the Connectivity service.

The `HttpDestination` library is released to the Maven central repository and can be used from [there](#). This is the recommended way to consume `HttpDestination` in the Neo environment.

Since on the Java Web Tomcat 8 runtime the `HttpDestination` functionality from Java Web is not part of the runtime, you must package some libraries with your application to use this functionality.

For **Maven** projects, you can easily achieve this by using the `pom.xml` file.

Reference HttpDestination

⚠ Caution

You must add both the `HttpDestination` dependencies and the external dependencies.

Add the `HttpDestination` library as a dependency in your `pom.xml` file as described in the [Maven Central Repository](#).

Sample Code

```
<dependency>
  <groupId>com.sap.cloud.connectivity</groupId>
  <artifactId>sap-cloud-connectivity-httpdestination</artifactId>
  <version>2.11.0</version>
</dependency>
```

External Dependencies

Some external dependencies are also required. Add the following dependencies to your `pom.xml` file.

Caution

Once you put those jars in your `web-inf/lib`, your application will **no longer work** on runtime 1 (OSGi runtime).

Sample Code


```
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpclient</artifactId>
  <version>4.5.10</version>
</dependency>
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpcore</artifactId>
  <version>4.4.12</version>
</dependency>
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpmime</artifactId>
  <version>4.5.2</version>
</dependency>
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpclient-cache</artifactId>
  <version>4.5.3</version>
</dependency>
<dependency>
  <groupId>commons-codec</groupId>
  <artifactId>commons-codec</artifactId>
  <version>1.13</version>
</dependency>
```


Note

Before the release of the `HttpDestination` library to the Maven central repository, SAP BTP provided a library based on Apache `HttpClient` 4.1.3, which had several limitations.

Consume `HttpDestination`

Features

- The recent versions of Apache `HttpClient` provide some new features (like async processing), better practices, etc.
- The SAP-specific library includes these main differences compared to the older versions:
 - Changed configuration options for `HttpClient` parameters
 - Usage of [ClosableHttpClient](#) 
 - Several deprecated methods and classes

For more information, see [release notes](#) .

PoolingConnectionManager Properties

With the new `HttpClient`, you cannot change the configuration of the pooling connection manager once it is created.

Therefore, if a client wants to configure the `maxTotalConnections` and `defaultMaxConnectionsPerRoute` properties of the connection manager, you must configure them as properties in the destination itself.

The Apache API defines the following methods for the `PoolingConnectionManager`:

Sample Code

```
setMaxTotal(int max); //Set the maximum number of total open connections. 20
by default.

setDefaultMaxPerRoute(int max); // Set the maximum number of concurrent
connections per route, which is 2 by default.
```

You can change the values by defining them in the *additional properties* of the HTTP destination:

- `MAX_TOTAL_CONNECTIONS` (`maxTotalConnections`)
- `DEFAULT_MAX_CONNECTIONS_PER_ROUTE` (`defaultMaxConnectionsPerRoute`)

See also [Create HTTP Destinations \[page 79\]](#).

HttpClient Class

As the `HttpClient` class implements `java.io.Closable`, a new class is introduced on SAP BTP side: `com.sap.core.connectivity.api.http.HttpDestinationClient`.

This class abstracts `org.apache.http.client.HttpClient`, and is the one that you should use.

1.4.1.2.2 HTTP Proxy for On-Premise Connectivity

The Connectivity service provides a standard HTTP Proxy for on-premise connectivity that is accessible by any application.

[Context \[page 153\]](#)

[Multitenancy Support \[page 153\]](#)

[Using the Proxy on Multi-Tenant VMs \[page 154\]](#)

[Using the Proxy on Single-Tenant VMs \[page 154\]](#)

Context

Proxy host and port are available as the environment variables `HC_OP_HTTP_PROXY_HOST` and `HC_OP_HTTP_PROXY_PORT`.

Note

- The HTTP Proxy provides a more flexible way to use on-premise connectivity via standard HTTP clients. It is not suitable for other protocols, such as `RFC` or `Mail`. HTTPS requests will not work as well.
- The previous alternative, that is, using on-premise connectivity via the existing HTTP Destination API, is still supported. For more information, see [HttpDestination Library \[page 148\]](#).

Caution

There is a limit of **8192** bytes for the size of the HTTP lines (for example, request line or header) that you send via the Connectivity service. If this limit is exceeded, you receive an HTTP error of type 4xx. This issue is usually caused by the size of the *path* + *query* string of the request.

Multitenancy Support

By default, all applications are started in multitenant mode. Such applications are responsible to propagate consumer subaccounts to the HTTP Proxy, using the header `SAP-Connectivity-ConsumerAccount`. This header is mandatory during the first request of each HTTP connection. HTTP connections are associated with one consumer subaccount and cannot be used with another subaccount. If the `SAP-Connectivity-ConsumerAccount` header is sent after the first request, and its value is different from the value in the first request, the Proxy will return HTTP response code 400.

Starting with SAP HANA Cloud Connector 2.9.0, it is possible to connect multiple Cloud Connectors to a subaccount as long as their location ID is different. Using the header `SAP-Connectivity-SCC-Location_ID` it is possible to specify the Cloud Connector over which the connection is opened. If this header is not specified, the connection will be opened to the Cloud Connector that is connected without any location ID. This is also the case for all Cloud Connector versions prior to 2.9.0.

If an application virtual machine (VM) is started for one consumer subaccount, this subaccount is known by the HTTP Proxy and the application may not send the **SAP-Connectivity-ConsumerAccount** header.

Using the Proxy on Multi-Tenant VMs

On multitenant VMs, applications are responsible to propagate consumer subaccount via **SAP-Connectivity-ConsumerAccount** header. The following example shows how this can be performed.

```
// TenantContext instance injection. It is used to get the consumer subaccount
name.
@Resource
public TenantContext tenantContext;
...

String proxyHost = System.getenv("HC_OP_HTTP_PROXY_HOST");
int proxyPort = Integer.parseInt(System.getenv("HC_OP_HTTP_PROXY_PORT"));

// set up the on-premise HTTP Proxy
HttpClient httpClient = new DefaultHttpClient();
httpClient.getParams().setParameter(ConnRoutePNames.DEFAULT_PROXY, new
HttpHost(proxyHost, proxyPort));

// insert the necessary headers in the request
HttpGet request = new HttpGet("http://virtualhost:1234");
request.addHeader("SAP-Connectivity-ConsumerAccount",
tenantContext.getTenant().getAccount().getId());

// execute the request
HttpResponse response = httpClient.execute(request);
```

Using the Proxy on Single-Tenant VMs

On single-tenant VMs, the consumer subaccount is known and subaccount propagation via header is not needed. The following example demonstrates this case.

```
String proxyHost = System.getenv("HC_OP_HTTP_PROXY_HOST");
int proxyPort = Integer.parseInt(System.getenv("HC_OP_HTTP_PROXY_PORT"));

// create HTTP client and insert the necessary headers in the request
HttpClient httpClient = new DefaultHttpClient();
httpClient.getParams().setParameter(ConnRoutePNames.DEFAULT_PROXY, new
HttpHost(proxyHost, proxyPort));
HttpGet request = new HttpGet("http://virtualhost:1234");

// execute the request
HttpResponse response = httpClient.execute(request);
```

Related Information

[Connectivity and Destination APIs \[page 136\]](#)

1.4.1.2.2.1 Principal Propagation Using HTTP Proxy

Context

The HTTP Proxy can forward the identity of an on-demand user to the Cloud Connector, and from there – to the back-end of the relevant on-premise system. In this way, on-demand users will no longer need to provide their identity every time they make connections to on-premise systems via one and the same Cloud Connector. To propagate the logged-in user, an application must use the `AuthenticationHeaderProvider` API to generate a header, which then embeds in the HTTP request to the on-premise system.

Restrictions

- IDPs used by applications protected by SAML2 have to be denoted as trustworthy for the Cloud Connector.
- Non-SAML2 protected applications have to be denoted themselves as trustworthy for the Cloud Connector.

Example

```
String proxyHost = System.getenv("HC_OP_HTTP_PROXY_HOST");
int proxyPort = Integer.parseInt(System.getenv("HC_OP_HTTP_PROXY_PORT"));
String account = System.getenv("HC_ACCOUNT");

// setup the on-premise HTTP proxy
HttpClient httpClient = new DefaultHttpClient();
httpClient.getParams().setParameter(ConnRoutePNames.DEFAULT_PROXY, new
    HttpHost(proxyHost, proxyPort));

// look up the connectivity authentication header provider resource called
"authHeaderProvider" (must be defined in web.xml)
Context ctx = new InitialContext();
AuthenticationHeaderProvider authHeaderProvider = (AuthenticationHeaderProvider)
    ctx.lookup("java:comp/env/authHeaderProvider");
// get header for principal propagation
AuthenticationHeader principalPropagationHeader =
    authHeaderProvider.getPrincipalPropagationHeader();

//insert the necessary headers in the request
HttpGet request = new HttpGet("http://virtualhost:1234");
request.addHeader(principalPropagationHeader.getName(),
    principalPropagationHeader.getValue());
request.addHeader("SAP-Connectivity-ConsumerAccount", account);
```

```
// execute the request
HttpResponse response = httpClient.execute(request);
```

Note

You can also apply dependency injection by using the `@Resource` annotation.

Related Information

[AuthenticationHeaderProvider API \[page 142\]](#)

[HTTP Proxy for On-Premise Connectivity \[page 153\]](#)

1.4.1.2.3 Use Cases

Overview

The Connectivity service enables access to remote services running either on the Internet or in an on-premise network.

Use Cases


The examples in this section show how you can make connections to Internet services and on-premise networks:

[Consume Internet Services \(Java Web or Java EE 6 Web Profile\) \[page 156\]](#)

[Consume Backend Systems \(Java Web or Java EE 6 Web Profile\) \[page 171\]](#)

1.4.1.2.3.1 Consume Internet Services (Java Web or Java EE 6 Web Profile)

Context

This example demonstrates consumption of Internet services using [Apache HTTP Client](#) . The example also shows how a connectivity-enabled Web application can be deployed on a local server and on the cloud.

The servlet code, the `web.xml` content, and the destination file (**outbound-internet-destination**) used in this example are mapped to the connectivity sample project located in `<SDK_location>/samples/connectivity`. You can directly import this sample in your Eclipse IDE. For more information, see .

Go through the relevant steps:

1. [Create a Dynamic Web Project \[page 197\]](#)
2. [Create a Sample Servlet \[page 198\]](#)
3. [Test the Connectivity-Enabled Web Application Locally \[page 161\]](#)
4. [Deploy the Connectivity-Enabled Web Application on the Cloud \[page 162\]](#)

Prerequisites

You have downloaded and set up your Eclipse IDE, SAP BTP Tools for Java, and SDK.

For more information, see [Setting Up the Development Environment](#).

Note

You need to install **SDK for Java Web** or **SDK for Java EE 6 Web Profile**.

1. Create a Dynamic Web Project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. From the Eclipse main menu, choose **File > New > Dynamic Web Project** .
3. In the *Project name* field, enter **ConnectivityHelloWorld** .
4. In the *Target Runtime* pane, select the runtime you want to use to deploy the application. In this example, we choose **Java Web**.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to finalize the creation of your project.

New Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: ConnectivityHelloWorld

Project location

☒ Use default location

Location: C:\Users\August\ConnectivityHelloWorld Browse...

Target runtime

Java Web New Runtime...

Dynamic web module version

2.5

Configuration

Default Configuration for Java Web Modify...

A good starting point for working with Java Web runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

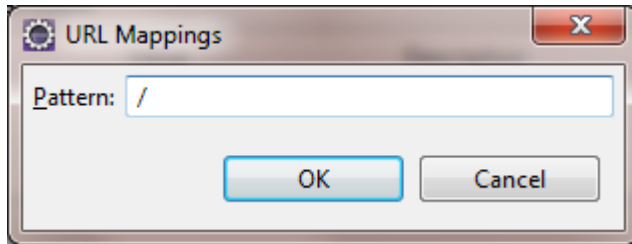
☐ Add project to an EAR

EAR project name: ConnectivityHelloWorldEAR New Project...

? < Back Next > Finish Cancel

2. Create a Sample Servlet

1. From the *ConnectivityHelloWorld* context menu, choose **New** > **Servlet**.
2. Enter **hello** as the Java package and **ConnectivityServlet** as the *Class name* and choose **Next**.
3. In the *URL mappings* field, select **/ConnectivityServlet** and choose **Edit**.
4. In the *Pattern* field, replace the current value with just **/**. In this way, the servlet will be mapped as a welcome page for the application.



5. Choose *Finish* so that the `ConnectivityServlet.java` servlet is created and opened in the Java editor.
6. Go to **ConnectivityHelloWorld** > **WebContent** > **WEB-INF** and open the `web.xml` file.
7. Choose the *Source* tab page.
8. Add the following code block to the `<web-app>` element:

```
<resource-ref>
    <res-ref-name>outbound-internet-destination</res-ref-name>
    <res-type>com.sap.core.connectivity.api.http.HttpDestination</res-type>
</resource-ref>
```

Note

The value of the `<res-ref-name>` element in the `web.xml` file should match the name of the destination that you want to be retrieved at runtime. In this case, the destination name is **outbound-internet-destination**.

9. Replace the entire servlet class with the following one to make use of the destination API. The destination API is visible by default for cloud applications and must not be added explicitly to the application class path.

```
package com.sap.cloud.sample.connectivity;
import java.io.IOException;
import java.io.InputStream;
import static java.net.HttpURLConnection.HTTP_OK;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.sap.core.connectivity.api.DestinationFactory;
import com.sap.core.connectivity.api.http.HttpDestination;
/**
 * Servlet class making HTTP calls to specified HTTP destinations.
 * Destinations are used in the following exemplary connectivity
scenarios:<br>
 * - Connecting to an outbound Internet resource using HTTP destinations<br>
 * - Connecting to an on-premise backend using on-premise HTTP
destinations,<br>
 * where the destinations could have no authentication or basic
authentication.<br>
 *
 * * NOTE: The Connectivity
                                service API is located under
 * <code>com.sap.core.connectivity.api</code>. The old API under
```

```

* <code>com.sap.core.connectivity.httpdestination.api</code> has been
deprecated.
*/
public class ConnectivityServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final int COPY_CONTENT_BUFFER_SIZE = 1024;
    private static final Logger LOGGER =
LoggerFactory.getLogger(ConnectivityServlet.class);
    /** {@inheritDoc} */
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpClient httpClient = null;
        String destinationName = request.getParameter("destname");
        try {
            // Get HTTP destination
            Context ctx = new InitialContext();
            HttpDestination destination = null;
            if (destinationName != null) {
                DestinationFactory destinationFactory = (DestinationFactory)
ctx.lookup(DestinationFactory.JNDI_NAME);
                destination = (HttpDestination)
destinationFactory.getDestination(destinationName);
            } else {
                // The default request to the Servlet will use outbound-
internet-destination
                destinationName = "outbound-internet-destination";
                destination = (HttpDestination) ctx.lookup("java:comp/env/"
+ destinationName);
            }
            // Create HTTP client
            httpClient = destination.createHttpClient();
            // Execute HTTP request
            HttpGet httpGet = new HttpGet();
            HttpResponse httpResponse = httpClient.execute(httpGet);
            // Check response status code
            int statusCode = httpResponse.getStatusLine().getStatusCode();
            if (statusCode != HTTP_OK) {
                throw new ServletException("Expected response status code is
200 but it is " + statusCode + " .");
            }
            // Copy content from the incoming response to the outgoing
response
            HttpEntity entity = httpResponse.getEntity();
            if (entity != null) {
                InputStream instream = entity.getContent();
                try {
                    byte[] buffer = new byte[COPY_CONTENT_BUFFER_SIZE];
                    int len;
                    while ((len = instream.read(buffer)) != -1) {
                        response.getOutputStream().write(buffer, 0, len);
                    }
                } catch (IOException e) {
                    // In case of an IOException the connection will be
released
                    // back to the connection manager automatically
                    throw e;
                } catch (RuntimeException e) {
                    // In case of an unexpected exception you may want to
abort
                    // the HTTP request in order to shut down the underlying
// connection immediately.
                    httpGet.abort();
                    throw e;
                } finally {
                    // Closing the input stream will trigger connection
release
                    try {

```

```

        instream.close();
    } catch (Exception e) {
        // Ignore
    }
}
} catch (NamingException e) {
    // Lookup of destination failed
    String errorMessage = "Lookup of destination failed with reason: "
        + e.getMessage()
        + ". See "
        + "logs for details. Hint: Make sure to have the
destination "
        + destinationName + " configured.";
    LOGGER.error("Lookup of destination failed", e);
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
        errorMessage);
} catch (Exception e) {
    // Connectivity operation failed
    String errorMessage = "Connectivity operation failed with reason:
"
        + e.getMessage()
        + ". See "
        + "logs for details. Hint: Make sure to have an HTTP
proxy configured in your "
        + "local Eclipse environment in case your environment
uses "
        + "an HTTP proxy for the outbound Internet "
        + "communication.";
    LOGGER.error("Connectivity operation failed", e);
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
        errorMessage);
} finally {
    // When HttpClient instance is no longer needed, shut down the
connection manager to ensure immediate
// deallocation of all system resources
    if (httpClient != null) {
        httpClient.getConnectionManager().shutdown();
    }
}
}
}

```

Note

The given servlet can run with different destination scenarios, for which user should specify the destination name as a requested parameter in the calling URL. In this case, the destination name should be <applicationURL>/?destname=outbound-internet-destination. Nevertheless, your servlet can still run even without specifying the destination name for this outbound scenario.

10. Save the Java editor and make sure the project compiles without errors.

3. Test the Connectivity-Enabled Web Application Locally

Caution

- If you use SDK for Java Web, we only recommend that you create a destination before deploying the application.
- If you use SDK for Java EE 6 Web Profile, you must create a destination before deploying the application.

1. In the context menu of the *Servers* view, choose **New > Server**.
2. Expand the *SAP* node, select *Java Web Server* and choose *Finish*.
3. A new server *Java Web Server [Stopped, Synchronized]* appears on the *Servers* tab page. Also, a *Servers* folder is created and appears in the navigation tree of the IDE. It contains configurable folders and files you can use, for example, to change your HTTP or JMX ports.
4. If you work behind a proxy server, you need to configure your proxy setting as follows:
 - In the *Servers* view, double-click the added server to open the editor.
 - Click the *Open Launch Configuration* link.
 - Choose the *(x)=Arguments* tab page.
 - In the *VM Arguments* box, add the following row:


```
-Dhttp.proxyHost=<your_proxy_host> -Dhttp.proxyPort=<your_proxy_port>
-Dhttps.proxyHost=<your_proxy_host> -Dhttps.proxyPort=<your_proxy_port>
```
 - Choose *OK*.
5. Go to the *Connectivity* tab page of your local server. Create a destination with the name **outbound-internet-destination**, and configure it so it can be consumed by the application at runtime. For more information, see [Configure Destinations from the Eclipse IDE \[page 63\]](#).
For the sample destination to work properly, the following properties need to be configured:


```
Name=outbound-internet-destination
Type=HTTP
URL=http://sap.com/index.html
Authentication=NoAuthentication
```
6. From the *ConnectivityServlet.java* editor's context menu, choose **Run As > Run on Server**.
7. Make sure that the *Choose an existing server* option is selected and choose *Java Web Server*.
8. Choose *Finish*.
The server is now started, displayed as *Java Web Server [Started, Synchronized]* in the *Servers* view.

Result:

The internal Web browser opens with the expected output of the connectivity-enabled Web application.

4. Deploy the Connectivity-Enabled Web Application on the Cloud

1. In the context menu of the *Servers* view, choose **New > Server**.
2. Choose SAP BTP as the type of server you want to create and choose *Next*.
3. For *Server's host name*, specify the region host depending on your subaccount type. For more information, see [Regions and Hosts Available for the Neo Environment](#).
4. Choose *Next*.
5. On the *New Server* wizard page, enter your application and subaccount name. Note that only lowercase Latin letters and digits are allowed.

Note

The application name should be unique enough to allow your deployed application to be easily identified in SAP BTP cockpit.

- Enter your subaccount name, e-mail or user name, and password.

- Choose *Finish*.
- A new server `<application>.<subaccount> [Stopped]` appears in the *Servers* view.
- Go to the *Connectivity* tab page of the server, create a destination with the name **outbound-internet-destination**, and configure it using the following properties:

```
Name=outbound-internet-destination
Type=HTTP
URL=http://sap.com/index.html
Authentication=NoAuthentication
ProxyType=Internet
```

- From the *ConnectivityServlet.java* editor's context menu, choose **Run As** **Run on Server**.
- Make sure that the *Choose an existing server* option is selected and choose **<Server_host_name>** **> <Server_name>**.
- Choose *Finish*.

Result:

The internal Web browser opens with the URL pointing to SAP BTP and displaying the expected output of the connectivity-enabled Web application.

1.4.1.2.3.2 Consume Internet Services (Java Web Tomcat 7)

Context

This example demonstrates consumption of Internet services using `HttpURLConnection`. The example also shows how a connectivity-enabled Web application can be deployed on a local server and on the cloud.

The servlet code, the `web.xml` content, and the destination file (**outbound-internet-destination**) used in this example are mapped to the connectivity sample project located in `<SDK_location>/samples/connectivity`. You can directly import this sample in your Eclipse IDE. For more information, see .

Go through the relevant steps:

1. [Create a Dynamic Web Project \[page 197\]](#)
2. [Create a Sample Servlet \[page 198\]](#)
3. [Test the Connectivity-Enabled Web Application Locally \[page 168\]](#)
4. [Deploy the Connectivity-Enabled Web Application on the Cloud \[page 169\]](#)

Prerequisites


You have downloaded and set up your Eclipse IDE, SAP BTP Tools for Java, and SDK.

For more information, see [Setting Up the Development Environment](#).

Note

You need to install **SDK for Java Web Tomcat 7**.

1. Create a Dynamic Web Project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. From the Eclipse main menu, choose **File > New > Dynamic Web Project** .
3. In the *Project name* field, enter **ConnectivityHelloWorld** .
4. In the *Target Runtime* pane, select **Java Web Tomcat 7** as the runtime you want to use to deploy the application.

5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to finalize the creation of your project.

New Dynamic Web Project

Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location
☒ Use default location
Location:

Target runtime

Dynamic web module version

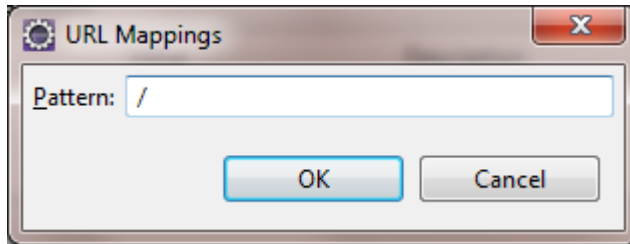
Configuration

A good starting point for working with Java Web Tomcat 7 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership
☐ Add project to an EAR
EAR project name:

2. Create a Sample Servlet

1. From the *ConnectivityHelloWorld* context menu, choose **New** > *Servlet*.
2. Enter **hello** as the Java package and **ConnectivityServlet** as the *Class name* and choose *Next*.
3. In the *URL mappings* field, select */ConnectivityServlet* and choose *Edit*.
4. In the *Pattern* field, replace the current value with just */*. In this way, the servlet will be mapped as a welcome page for the application.



5. Choose *Finish* so that the `ConnectivityServlet.java` servlet is created and opened in the Java editor.
6. Go to **ConnectivityHelloWorld > WebContent > WEB-INF** and open the `web.xml` file.
7. Choose the *Source* tab page.
8. To consume connectivity configuration using JNDI, you need to define the `ConnectivityConfiguration` API as a resource in the `web.xml` file. Below is an example of a `ConnectivityConfiguration` resource, named **connectivityConfiguration**.

```
<resource-ref>
  <res-ref-name>connectivityConfiguration</res-ref-name>
  <res-
type>com.sap.core.connectivity.api.configuration.ConnectivityConfiguration</
res-type>
</resource-ref>
```

9. Replace the entire servlet class with the following one to make use of the destination API. The destination API is visible by default for cloud applications and must not be added explicitly to the application class path.

```
package com.sap.cloud.sample.connectivity;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.InetSocketAddress;
import java.net.Proxy;
import java.net.URL;

import javax.annotation.Resource;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.sap.cloud.account.TenantContext;
import com.sap.core.connectivity.api.configuration.ConnectivityConfiguration;
import com.sap.core.connectivity.api.configuration.DestinationConfiguration;

/**
 * Servlet class making http calls to specified http destinations.
 * Destinations are used in the following example connectivity scenarios:<br>
 * - Connecting to an outbound Internet resource using HTTP destinations<br>
 * - Connecting to an on-premise backend using on premise HTTP
 destinations,<br>
 * where the destinations have no authentication.<br>
 */
public class ConnectivityServlet extends HttpServlet {
    @Resource
    private TenantContext tenantContext;
```

```

        private static final long serialVersionUID = 1L;
        private static final int COPY_CONTENT_BUFFER_SIZE = 1024;
        private static final Logger LOGGER =
LoggerFactory.getLogger(ConnectivityServlet.class);

        private static final String ON_PREMISE_PROXY = "OnPremise";

        /** {@inheritDoc} */
        @Override
        public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            HttpURLConnection urlConnection = null;
            String destinationName = request.getParameter("destname");

            // The default request to the Servlet will use outbound-internet-
destination
            if (destinationName == null) {
                destinationName = "outbound-internet-destination";
            }

            try {
                // Look up the connectivity configuration API
                Context ctx = new InitialContext();
                ConnectivityConfiguration configuration
= (ConnectivityConfiguration) ctx.lookup("java:comp/env/
connectivityConfiguration");

                // Get destination configuration for "destinationName"
                DestinationConfiguration destConfiguration =
configuration.getConfiguration(destinationName);
                if (destConfiguration == null) {

response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                    String.format("Destination %s is not found. Hint:
Make sure to have the destination configured.", destinationName));
                    return;
                }

                // Get the destination URL
                String value = destConfiguration.getProperty("URL");
                URL url = new URL(value);

                String proxyType = destConfiguration.getProperty("ProxyType");
                Proxy proxy = getProxy(proxyType);

                urlConnection = (HttpURLConnection) url.openConnection(proxy);

                // Insert the required header in the request for on-premise
destinations
                injectHeader(urlConnection, proxyType);

                // Copy content from the incoming response to the outgoing
response
                InputStream instream = urlConnection.getInputStream();
                OutputStream outstream = response.getOutputStream();
                copyStream(instream, outstream);
            } catch (Exception e) {
                // Connectivity operation failed
                String errorMessage = "Connectivity operation failed with reason:
"
                + e.getMessage()
                + ". See "
                + "logs for details. Hint: Make sure to have an HTTP
proxy configured in your "
                + "local environment in case your environment uses "
                + "an HTTP proxy for the outbound Internet "
                + "communication.";
            }

```

```

        LOGGER.error("Connectivity operation failed", e);
        response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
            errorMessage);
    }

    private Proxy getProxy(String proxyType) {
        String proxyHost = null;
        int proxyPort;

        if (ON_PREMISE_PROXY.equals(proxyType)) {
            // Get proxy for on-premise destinations
            proxyHost = System.getenv("HC_OP_HTTP_PROXY_HOST");
            proxyPort =
Integer.parseInt(System.getenv("HC_OP_HTTP_PROXY_PORT"));
        } else {
            // Get proxy for internet destinations
            proxyHost = System.getProperty("http.proxyHost");
            proxyPort =
Integer.parseInt(System.getProperty("http.proxyPort"));
        }
        return new Proxy(Proxy.Type.HTTP, new InetSocketAddress(proxyHost,
proxyPort));
    }

    private void injectHeader(HttpURLConnection urlConnection, String
proxyType) {
        if (ON_PREMISE_PROXY.equals(proxyType)) {
            // Insert header for on-premise connectivity with the consumer
subaccount name
            urlConnection.setRequestProperty("SAP-Connectivity-
ConsumerAccount", tenantContext.getAccountName());
        }
    }

    private void copyStream(InputStream inStream, OutputStream outStream)
throws IOException {
        byte[] buffer = new byte[COPY_CONTENT_BUFFER_SIZE];
        int len;
        while ((len = inStream.read(buffer)) != -1) {
            outStream.write(buffer, 0, len);
        }
    }
}

```

📌 Note

The given servlet can run with different destination scenarios, for which user should specify the destination name as a requested parameter in the calling URL. In this case, the destination name should be <applicationURL>/?destname=outbound-internet-destination. Nevertheless, your servlet can still run even without specifying the destination name for this outbound scenario.

10. Save the Java editor and make sure the project compiles without errors.

3. Test the Connectivity-Enabled Web Application Locally

📌 Note

We recommend but not obligate that you create a destination before deploying the application.

1. In the context menu of the **Servers** view, choose **New > Server**.
2. Expand the **SAP** node, select **Java Web Tomcat 7 Server** and choose **Finish**.
3. A new server **Java Web Tomcat 7 Server [Stopped, Synchronized]** appears on the **Servers** tab page.

Also, a **Servers** folder is created and appears in the navigation tree of the IDE. It contains configurable folders and files you can use, for example, to change your HTTP or JMX ports.

4. If you work behind a proxy server, you need to configure your proxy setting as follows:

- In the **Servers** view, double-click the added server to open the editor.
- Click the **Open Launch Configuration** link.
- Choose the **(x)=Arguments** tab page.
- In the **VM Arguments** box, add the following row:

```
-Dhttp.proxyHost=<your_proxy_host> -Dhttp.proxyPort=<your_proxy_port>
-Dhttps.proxyHost=<your_proxy_host> -Dhttps.proxyPort=<your_proxy_port>
```

- Choose **OK**.
5. Go to the **Connectivity** tab page of your local server, create a destination with the name **outbound-internet-destination**, and configure it so it can be consumed by the application at runtime. For more information, see [Configure Destinations from the Eclipse IDE \[page 63\]](#).

For the sample destination to work properly, the following properties need to be configured:

```
Name=outbound-internet-destination
Type=HTTP
URL=http://sap.com/index.html
Authentication=NoAuthentication
```

6. From the **ConnectivityServlet.java** editor's context menu, choose **Run As > Run on Server**.
7. Make sure that the **Choose an existing server** option is selected and choose **Java Web Tomcat 7 Server**.
8. Choose **Finish**.
The server is now started, displayed as **Java Web Tomcat 7 Server [Started, Synchronized]** in the **Servers** view.

Result:

The internal Web browser opens with the expected output of the connectivity-enabled Web application.

4. Deploy the Connectivity-Enabled Web Application on the Cloud

1. In the context menu of the **Servers** view, choose **New > Server**.
2. Choose **SAP Cloud Platform** as the type of server you want to create and choose **Next**.
3. For **Server's host name**, specify the region host depending on your subaccount type. For more information, see [Regions and Hosts Available for the Neo Environment](#).
4. Choose **Next**.
5. On the **New Server** wizard page, enter your application and subaccount name. Note that only lowercase Latin letters and digits are allowed.

Note

The application name should be unique enough to allow your deployed application to be easily identified in SAP BTP cockpit.

6. Enter your subaccount name, e-mail or user name, and password.

New Server

SAP BTP Application

Specify application name and subaccount data

Application name: myapp

Runtime: Automatic

Subaccount name: Java EE 6 Web Profile

User name: p1234567890

Password:

☒ Save password (could trigger secure storage login)

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

7. Choose *Finish*.
8. A new server `<application>.<subaccount> [Stopped]` appears in the *Servers* view.
9. Go to the *Connectivity* tab page of the server. Create a destination with the name **outbound-internet-destination**, and configure it using the following properties:

```
Name=outbound-internet-destination
Type=HTTP
URL=http://sap.com/index.html
Authentication=NoAuthentication
ProxyType=Internet
```

10. From the `ConnectivityServlet.java` editor's context menu, choose **Run As** **Run on Server**.
11. Make sure that the *Choose an existing server* option is selected and choose **<Server_host_name>** **<Server_name>**.
12. Choose *Finish*.

Result:

The internal Web browser opens with the URL pointing to SAP BTP and displaying the expected output of the connectivity-enabled Web application.

1.4.1.2.3.3 Consume Backend Systems (Java Web or Java EE 6 Web Profile)

Context

This example demonstrates how a sample Web application consumes a backend system via HTTP(S) by using the Connectivity service. For simplicity, instead of using a real backend system, we use a second sample Web application containing `BackendServlet`. It mimics the backend system and can be called via HTTP(S).

The servlet code, the `web.xml` content, and the destination files (**backend-no-auth-destination** and **backend-basic-auth-destination**) used in this example are mapped to the connectivity sample project located in `<SDK_location>/samples/connectivity`. You can directly import this sample in your Eclipse IDE. For more information, see .

The example includes the following sections:

1. [Set Up Application as a Backend System \[page 172\]](#)
2. [Create a Dynamic Web Project \[page 197\]](#)
3. [Create a Sample Servlet \[page 175\]](#)
4. [Deploy the Application \[page 199\]](#)
5. [Configure the Destination in the Cloud \[page 180\]](#)

Connectivity User Roles

In the on-demand to on-premise connectivity end-to-end scenario, different user roles are involved. The particular steps for the relevant roles are described below:

- **IT Administrator** - Sets up and configures the Cloud Connector. Scenario steps:
 1. Downloads the Cloud Connector from <https://tools.hana.ondemand.com/#cloud>
 2. Installs the connector.
 3. Establishes an SSL tunnel from the connector to an SAP BTP subaccount.
 4. Configures the exposed backend systems and resources.
- **Application Developer** - Develops Web applications using destinations. Scenario steps:
 1. Installs the Eclipse IDE, SAP BTP Tools for Java, and SDK.
 2. Develops a Java EE application using the destination API.

3. Configures connectivity destinations as resources in the `web.xml` file.
 4. Configures connectivity destinations via the SAP BTP server adapter in Eclipse IDE.
 5. Deploys the Java EE application locally and on the cloud.
- **Subaccount Operator** - Deploys Web applications, configures their destinations, and conducts tests.
Scenario steps:
 1. Obtains a ready Java EE application WAR file.
 2. Deploys the Java EE application to an SAP BTP subaccount.
 3. Uploads the connectivity destination configuration via the console client.
 4. Tests the Java EE application on a local server and deploys it again to an SAP BTP subaccount.

For more information, see [Cloud Connector \[page 232\]](#).

Prerequisites

- You have downloaded and configured the Cloud Connector. For more information, see [Cloud Connector \[page 232\]](#).
- You have downloaded and set up your Eclipse IDE, SAP BTP Tools for Java, and SDK.
For more information, see [Setting Up the Development Environment](#).

Note

You need to install **SDK for Java Web** or **SDK for Java EE 6 Web Profile**.

1. Set Up Application as a Backend System

This example uses a Web application that responds to a request with a ping as a sample backend system. The Connectivity service supports HTTP and HTTPS as protocols and provides an easy way to consume REST-based Web services.

To set up the sample application as a backend system, see [Set Up an Application as a Sample Backend System \[page 190\]](#).


→ Tip

Instead of the sample backend system provided in this example, you can use other systems to be consumed through REST-based Web services.

Once the backend application is running on your local Tomcat, you need to configure the ping service, provided by the application, in your installed Cloud Connector. This is required since the Cloud Connector only allows access to trusted backend services. To do this, follow the steps below:

1. Open the Cloud Connector and from the **Content** navigation (in left), choose **Access Control**.
2. Under **Mapping Virtual To Internal System**, choose the **Add** button and define an entry as shown on the following screenshot. The **Internal Host** must be the physical host name of the machine on which the Tomcat of the backend application is running.

Add System Mapping

 **No resources are accessible unless explicitly allowed**

Virtual Host: *

Virtual Port: *

Internal Host:

Internal Port:

Protocol:

Principal Type:

Back-end Type: *

SNC Partner Name:

☒ Check availability of internal host when saving

Note









This step shows the procedure and screenshot for Cloud Connector versions prior to 2.9. For Cloud Connector versions as of 2.9.0, follow the steps in [Configure Access Control \(HTTP\) \[page 342\]](#) and enter the values shown in the screenshot above.

- Choose [Save](#). The newly mapped system appears in the table.
- Click it. A new table, *Resources Accessible On <host>:<port>*, opens below.

Note

For Cloud Connector versions as of 2.9.0, follow the steps in [Configure Access Control \(HTTP\) \[page 342\]](#), section *Limiting the Accessible Services for HTTP(S)*, and enter the values as shown in the next step.

- Specify the URL paths `/BackendAppHttpBasicAuth` and `/BackendAppHttpNoAuth` as accessible resources, as shown on the screenshot below. When defining the paths, make sure you have selected the *Path and all sub-paths* option.

 Add...  Edit...  Enable  Disable  Delete			
	State	URL Path	Access Policy
		<code>/BackendAppHttpBasicAuth</code>	Path and all sub-paths
		<code>/BackendAppHttpNoAuth</code>	Path and all sub-paths

Note

In case you use SDK with version equal to or lower than **1.44.0.1** (Java Web) and **2.24.13** (Java EE 6 Web Profile), you should find the WAR files in directory `<SDK_location>/tools/samples/connectivity/onpremise`, under the names **PingAppHttpNoAuth.war** and **PingAppHttpBasicAuth.war**. Also, the URL paths should be `/PingAppHttpBasicAuth` and `/PingAppHttpNoAuth`.

2. Create a Dynamic Web Project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. From the Eclipse main menu, choose **File** **>** **New** **>** *Dynamic Web Project*.
3. In the *Project name* field, enter **ConnectivityHelloWorld**.
4. In the *Target Runtime* pane, select the runtime you want to use to deploy the application. In this example, we choose **Java Web**.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to finalize the creation of your project.

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: ConnectivityHelloWorld

Project location

☒ Use default location

Location: C:\Users\August\ConnectivityHelloWorld

Target runtime

Java Web

Dynamic web module version

2.5

Configuration

Default Configuration for Java Web

A good starting point for working with Java Web runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

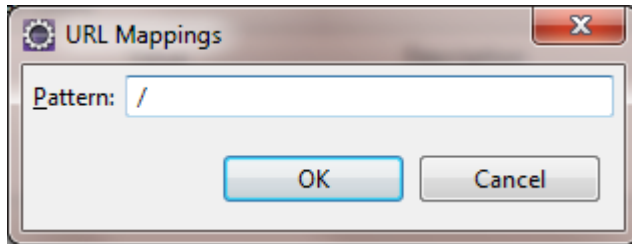
☐ Add project to an EAR

EAR project name: ConnectivityHelloWorldEAR

< Back Next > Finish Cancel

3. Create a Sample Servlet

1. From the *ConnectivityHelloWorld* context menu, choose **New** > **Servlet**.
2. Enter **hello** as the *Java package* and **ConnectivityServlet** as the *Class name* and choose **Next**.
3. In the *URL mappings* field, select **/ConnectivityServlet** and choose **Edit**.
4. In the *Pattern* field, replace the current value with just **/**. In this way, the servlet will be mapped as a welcome page for the application.



5. Choose *Finish* so that the *ConnectivityServlet.java* servlet is created and opened in the Java editor.
6. Go to **ConnectivityHelloWorld > WebContent > WEB-INF** and open the *web.xml* file.
7. Add the following code block to the `<web-app>` element, respectively:

```
<resource-ref>
  <res-ref-name>outbound-internet-destination</res-ref-name>
  <res-type>com.sap.core.connectivity.api.http.HttpDestination</res-type>
</resource-ref>
```

```
<resource-ref>
  <res-ref-name>connectivity/DestinationFactory</res-ref-name>
  <res-type>com.sap.core.connectivity.api.DestinationFactory</res-type>
</resource-ref>
```

Note

- Destinations **backend-no-auth-destination** and **backend-basic-auth-destination** will be looked-up via DestinationFactory JNDI lookup. For more information, see [HttpDestination Library \[page 148\]](#).
- In case you use destinations as resource reference, the value of the `<res-ref-name>` element in the *web.xml* file should match the name of the destination that you want to be retrieved at runtime. In this case, the destination name is **outbound-internet-destination**.

8. Replace the entire servlet class to make use of the destination API. The destination API is visible by default for cloud applications and must not be added explicitly to the application class path.

```
package com.sap.cloud.sample.connectivity;

import java.io.IOException;
import java.io.InputStream;

import static java.net.HttpURLConnection.HTTP_OK;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.sap.core.connectivity.api.http.HttpDestination;
import com.sap.core.connectivity.api.DestinationFactory;
```

```

/**
 * Servlet class making HTTP calls to specified HTTP destinations.
 * Destinations are used in the following exemplary connectivity
scenarios:<br>
 * - Connecting to an outbound Internet resource using HTTP destinations<br>
 * - Connecting to an on-premise backend using on-premise HTTP
destinations,<br>
 * where the destinations could have no authentication or basic
authentication.<br>
 *
 * * NOTE: The Connectivity
                                service API is located under
 * <code>com.sap.core.connectivity.api</code>. The old API under
 * <code>com.sap.core.connectivity.httpdestination.api</code> has been
deprecated.
 */
public class ConnectivityServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final int COPY_CONTENT_BUFFER_SIZE = 1024;
    private static final Logger LOGGER =
LoggerFactory.getLogger(ConnectivityServlet.class);

    /** {@inheritDoc} */
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpClient httpClient = null;
        String destinationName = request.getParameter("destname");
        try {
            // Get HTTP destination
            Context ctx = new InitialContext();
            HttpDestination destination = null;
            if (destinationName != null) {
                DestinationFactory destinationFactory = (DestinationFactory)
ctx.lookup(DestinationFactory.JNDI_NAME);
                destination = (HttpDestination)
destinationFactory.getDestination(destinationName);
            } else {
                // The default request to the Servlet will use outbound-
internet-destination
                destinationName = "outbound-internet-destination";
                destination = (HttpDestination) ctx.lookup("java:comp/env/"
+ destinationName);
            }

            // Create HTTP client
            httpClient = destination.createHttpClient();

            // Execute HTTP request
            HttpGet httpGet = new HttpGet();
            HttpResponse httpResponse = httpClient.execute(httpGet);

            // Check response status code
            int statusCode = httpResponse.getStatusLine().getStatusCode();
            if (statusCode != HTTP_OK) {
                throw new ServletException("Expected response status code is
200 but it is " + statusCode + " .");
            }

            // Copy content from the incoming response to the outgoing
response
            HttpEntity entity = httpResponse.getEntity();
            if (entity != null) {
                InputStream instream = entity.getContent();
                try {
                    byte[] buffer = new byte[COPY_CONTENT_BUFFER_SIZE];
                    int len;
                    while ((len = instream.read(buffer)) != -1) {

```

```

        response.getOutputStream().write(buffer, 0, len);
    }
    } catch (IOException e) {
        // In case of an IOException the connection will be
released
        // back to the connection manager automatically
        throw e;
    } catch (RuntimeException e) {
        // In case of an unexpected exception you may want to
abort
        // the HTTP request in order to shut down the underlying
        // connection immediately.
        httpGet.abort();
        throw e;
    } finally {
        // Closing the input stream will trigger connection
release
        try {
            instream.close();
        } catch (Exception e) {
            // Ignore
        }
    }
}
} catch (NamingException e) {
    // Lookup of destination failed
    String errorMessage = "Lookup of destination failed with reason: "
        + e.getMessage()
        + ". See "
        + "logs for details. Hint: Make sure to have the
destination "
        + destinationName + " configured.";
    LOGGER.error("Lookup of destination failed", e);
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
        errorMessage);
} catch (Exception e) {
    // Connectivity operation failed
    String errorMessage = "Connectivity operation failed with reason:
"
        + e.getMessage()
        + ". See "
        + "logs for details. Hint: Make sure to have an HTTP
proxy configured in your "
        + "local Eclipse environment in case your environment
uses "
        + "an HTTP proxy for the outbound Internet "
        + "communication.";
    LOGGER.error("Connectivity operation failed", e);
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
        errorMessage);
} finally {
    // When HttpClient instance is no longer needed, shut down the
connection manager to ensure immediate
    // deallocation of all system resources
    if (httpClient != null) {
        httpClient.getConnectionManager().shutdown();
    }
}
}
}

```

📌 Note

The given servlet can be run with different destination scenarios, for which user should specify the destination name as a requested parameter in the calling URL. In the case of on-premise connection to a backend system, the destination name should be either **backend-basic-auth-destination**

or **backend-no-auth-destination**, depending on the chosen authentication type scenario. For example: `<application_URL>/?destname=backend-no-auth-destination`

9. Save the Java editor and make sure the project compiles without errors.

4. Deploy the Application

⚠ Caution

- If you use SDK for Java Web, we just recommend that you create a destination before starting the application.
- If you use SDK for Java EE 6 Web Profile, you must create a destination **before** starting the application.

1. To deploy your Web application locally or on the cloud, follow the steps described in the respective pages:
2. Once the application is deployed successfully on a local server and on the cloud, the application issues an exception. This exception says that destination **backend-basic-auth-destination** or **backend-no-auth-destination** has not been specified yet:

```
HTTP Status 500 - Connectivity operation failed with reason: Destination with
name backend-no-auth-destination cannot be found. Make sure it is created and
configured.. See logs for details.
2014 01 10
08:11:01#+00#ERROR#com.sap.cloud.sample.connectivity.ConnectivityServlet##anon
ymous#http-bio-8041-exec-1##conngold#testsample#web#null#null#Connectivity
operation failed
com.sap.core.connectivity.api.DestinationNotFoundException: Destination with
name backend-no-auth-destination cannot be found. Make sure it is created and
configured.
    at
com.sap.core.connectivity.destinations.DestinationFactory.getDestination(Desti
nationFactory.java:20)
    at
com.sap.core.connectivity.cloud.destinations.CloudDestinationFactory.getDestin
ation(CloudDestinationFactory.java:28)
    at
com.sap.cloud.sample.connectivity.ConnectivityServlet.doGet(ConnectivityServle
t.java:50)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:735)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:848)
    at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFi
lterChain.java:305)
    at
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChai
n.java:210)
    at
com.sap.core.communication.server.CertValidatorFilter.doFilter(CertValidatorFi
lter.java:321)
    at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFi
lterChain.java:243)
    ...
```

3. As a next step, you need to configure **backend-no-auth-destination** or **backend-basic-auth-destination**.

For more information, see [HttpDestination Library \[page 148\]](#).


5. Configure the Destination in the Cloud

To configure the destination in SAP BTP, you need to use the virtual host name (**virtualpingbackend**) and port (**1234**) specified in one of the previous steps on the Cloud Connector's [Access Control](#) tab page.

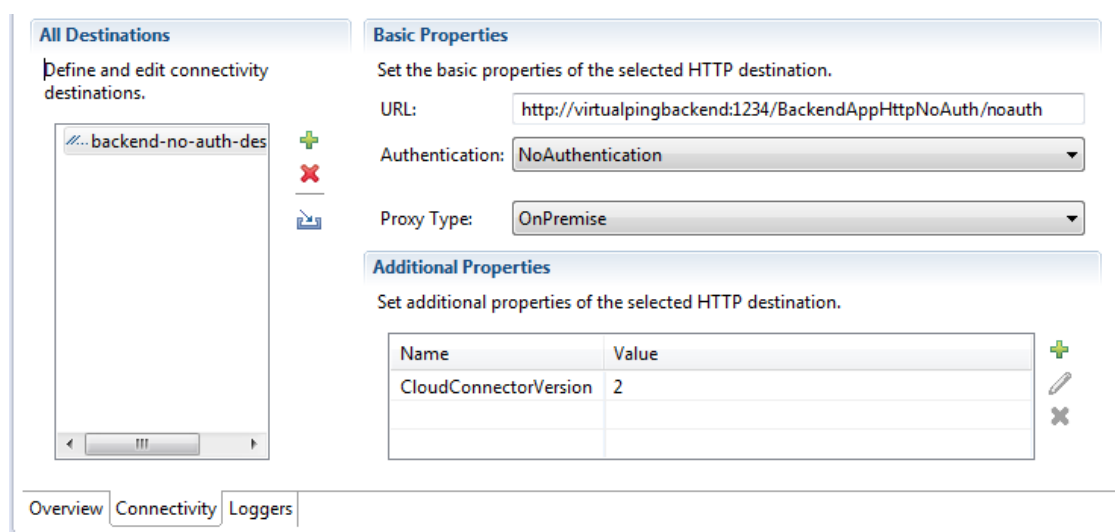
Note

On-premise destinations support HTTP connections only. Thus, when defining a destination in the SAP BTP cockpit, always enter the URL as **http://virtual.host:virtual.port**, even if the backend requires an HTTPS connection.

The connection from an SAP BTP application to the Cloud Connector (through the tunnel) is encrypted with TLS anyway. There is no need to “double-encrypt” the data. Then, for the leg from the Cloud Connector to the backend, you can choose between using HTTP or HTTPS. The Cloud Connector will establish an SSL/TLS connection to the backend, if you choose HTTPS.

1. In the Eclipse IDE, open the [Servers](#) view and double-click on `<application>.<subaccount>` to open the SAP BTP editor.
2. Open the [Connectivity](#) tab page.
3. In the [All Destinations](#) section, choose  to create a new destination with the name **backend-no-auth-destination** or **backend-basic-auth-destination**.
 - To connect with no authentication, use the following configuration:

```
Name=backend-no-auth-destination
Type=HTTP
URL=http://virtualpingbackend:1234/BackendAppHttpNoAuth/noauth
Authentication=NoAuthentication
ProxyType=OnPremise
CloudConnectorVersion=2
```



The screenshot shows the SAP BTP cockpit interface. On the left, the 'All Destinations' panel lists a destination named 'backend-no-auth-des'. On the right, the 'Basic Properties' panel is active, showing the configuration for this destination. The URL is 'http://virtualpingbackend:1234/BackendAppHttpNoAuth/noauth', Authentication is 'NoAuthentication', and Proxy Type is 'OnPremise'. Below this, the 'Additional Properties' section shows a table with 'CloudConnectorVersion' set to '2'.

Name	Value
CloudConnectorVersion	2

- To connect with basic authentication, use the following configuration:

```
Name=backend-basic-auth-destination
```



```
Type=HTTP
URL=http://virtualpingbackend:1234/BackendAppHttpBasicAuth/basic
Authentication=BasicAuthentication
User=pinguser
Password=pingpassword
ProxyType=OnPremise
CloudConnectorVersion=2
```

The screenshot shows the SAP BTP Connectivity editor with the 'Connectivity' tab selected. On the left, under 'All Destinations', a list contains 'backend-basic-auth'. The main area is divided into 'Basic Properties' and 'Additional Properties'.

Basic Properties:

- URL:
- Authentication:
- Basic authentication settings:
 - User:
 - Password:
- Proxy Type:

Additional Properties:

Name	Value
CloudConnectorVersion	2

At the bottom, there are tabs for 'Overview', 'Connectivity', and 'Loggers'.

4. Save the destination.
5. The [Connectivity](#) editor automatically saves the configuration in SAP BTP.
6. Call the URL that references the cloud application again in the Web browser. The application should now return the ping response.

1.4.1.2.3.4 Consume Backend Systems (Java Web Tomcat 7)

Context

This example demonstrates how a sample Web application consumes a backend system via HTTP(S) by using the Connectivity service. For simplicity, instead of using a real backend system, we use a second sample Web application containing `BackendServlet`. It mimics the backend system and can be called via HTTP(S).

The servlet code, the `web.xml` content, and the destination file ([backend-no-auth-destination](#)) used in this example are mapped to the connectivity sample project located in `<SDK_location>/samples/connectivity`. You can directly import this sample in your Eclipse IDE. For more information, see .

The example includes the following sections:

1. [Set Up Application as a Backend System \[page 183\]](#)
2. [Create a Dynamic Web Project \[page 197\]](#)
3. [Create a Sample Servlet \[page 185\]](#)
4. [Deploy the Application \[page 199\]](#)
5. [Configure the Destination in the Cloud \[page 189\]](#)

Connectivity User Roles

In the on-demand to on-premise connectivity end-to-end scenario, different user roles are involved. The particular steps for the relevant roles are described below:

- *IT Administrator* - Sets up and configures the Cloud Connector. Scenario steps:
 1. Downloads the Cloud Connector from <https://tools.hana.ondemand.com/#cloud>
 2. Installs the connector.
 3. Establishes an SSL tunnel from the connector to an SAP BTP.
 4. Configures the exposed backend systems and resources.
- *Application Developer* - Develops Web applications using destinations. Scenario steps:
 1. Installs the Eclipse IDE, SAP BTP Tools for Java, and SDK.
 2. Develops a Java EE application using the destination API.
 3. Configures connectivity destinations as resources in the `web.xml` file.
 4. Configures connectivity destinations via the SAP BTP server adapter in Eclipse IDE.
 5. Deploys the Java EE application locally and on the cloud.
- *Subaccount Operator* - Deploys Web applications, configures their destinations, and conducts tests. Scenario steps:
 1. Obtains a ready Java EE application WAR file.
 2. Deploys the Java EE application to an SAP BTP subaccount.
 3. Uploads the connectivity destination configuration via the console client.
 4. Tests the Java EE application on a local server and deploys it again to a SAP BTP subaccount.

For more information, see [Cloud Connector \[page 232\]](#).

Prerequisites

- You have downloaded and configured the Cloud Connector. For more information, see [Cloud Connector \[page 232\]](#).
- You have downloaded and set up your Eclipse IDE, SAP BTP Tools for Java, and SDK. For more information, see [Setting Up the Development Environment](#).

Note

You need to install **SDK for Java Web Tomcat 7**.

1. Set Up Application as a Backend System

This example uses a Web application that responds to a request with a ping as a sample backend system. The Connectivity service supports HTTP and HTTPS as protocols and provides an easy way to consume REST-based Web services.

To set up the sample application as a backend system, see [Set Up an Application as a Sample Backend System \[page 190\]](#).

→ Tip

Instead of the sample backend system provided in this example, you can use other systems to be consumed through REST-based Web services.

Once the backend application is running on your local Tomcat, you need to configure the ping service, provided by the application, in your installed Cloud Connector. This is required since the Cloud Connector only allows access to white-listed backend services. To do this, follow the steps below:

1. Open the Cloud Connector and from the [Content](#) navigation (in left), choose [Access Control](#).
2. Under [Mapping Virtual To Internal System](#), choose the [Add](#) button and define an entry as shown on the following screenshot. The [Internal Host](#) must be the physical host name of the machine on which the Tomcat of the backend application is running.

Add System Mapping

⚠ No resources are accessible unless explicitly allowed

Virtual Host: *

Virtual Port: *

Internal Host:

Internal Port:

Protocol:

Principal Type:

Back-end Type: *

SNC Partner Name:

☒ Check availability of internal host when saving

ⓘ Note








This step shows the procedure and screenshot for Cloud Connector versions prior to 2.9. For Cloud Connector versions as of 2.9.0, follow the steps in [Configure Access Control \(HTTP\) \[page 342\]](#) and enter the values shown in the screenshot above.

3. Choose [Save](#). The newly mapped system appears in the table.
4. Click it. A new table, *Resources Accessible On <host>:<port>*, opens below.

Note

For Cloud Connector versions as of 2.9.0, follow the steps in [Configure Access Control \(HTTP\) \[page 342\]](#), section *Limiting the Accessible Services for HTTP(S)*, and enter the values as shown in the next step.

5. Specify the URL paths `/BackendAppHttpBasicAuth` and `/BackendAppHttpNoAuth` as accessible resources, as shown on the screenshot below. When defining the paths, make sure you have selected the *Path and all sub-paths* option.

 Add...  Edit...  Enable  Disable  Delete			
	State	URL Path	Access Policy
		<code>/BackendAppHttpBasicAuth</code>	Path and all sub-paths
		<code>/BackendAppHttpNoAuth</code>	Path and all sub-paths

2. Create a Dynamic Web Project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. From the Eclipse main menu, choose **File** > **New** > **Dynamic Web Project**.
3. In the *Project name* field, enter **ConnectivityHelloWorld**.
4. In the *Target Runtime* pane, select **Java Web Tomcat 7** as the runtime you want to use to deploy the application.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to finalize the creation of your project.

New Dynamic Web Project

Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: ConnectivityHelloWorld

Project location
☒ Use default location
Location: C:\Users\August\ConnectivityHelloWorld Browse...

Target runtime
Java Web Tomcat 7 New Runtime...

Dynamic web module version
3.0

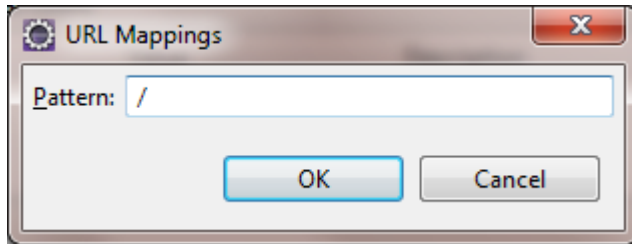
Configuration
Default Configuration for Java Web Tomcat 7 Modify...
A good starting point for working with Java Web Tomcat 7 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership
☐ Add project to an EAR
EAR project name: ConnectivityHelloWorldEAR New Project...

? < Back Next > Finish Cancel

3. Create a Sample Servlet

1. From the *ConnectivityHelloWorld* context menu, choose **New** > **Servlet**.
2. Enter **hello** as the *Java package* and **ConnectivityServlet** as the *Class name* and choose **Next**.
3. In the *URL mappings* field, select **/ConnectivityServlet** and choose **Edit**.
4. In the *Pattern* field, replace the current value with just **/**. In this way, the servlet will be mapped as a welcome page for the application.



5. Choose *Finish* so that the *ConnectivityServlet.java* servlet is created and opened in the Java editor.
6. Go to ► *ConnectivityHelloWorld* ► *WebContent* ► *WEB-INF* ► and open the *web.xml* file.
7. To consume connectivity configuration using JNDI, you need to define the ConnectivityConfiguration API as a resource in the *web.xml* file. Below is an example of a ConnectivityConfiguration resource, named **connectivityConfiguration**.

```
<resource-ref>
  <res-ref-name>connectivityConfiguration</res-ref-name>
  <res-
type>com.sap.core.connectivity.api.configuration.ConnectivityConfiguration</
res-type>
</resource-ref>
```

Note

Destination **backend-no-auth-destination** will be looked-up via ConnectivityConfiguration JNDI lookup. For more information, see [ConnectivityConfiguration API \[page 140\]](#).

8. Replace the entire servlet class to make use of the configuration API. The configuration API is visible by default for cloud applications and must not be added explicitly to the application class path.

```
package com.sap.cloud.sample.connectivity;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.InetSocketAddress;
import java.net.Proxy;
import java.net.URL;

import javax.annotation.Resource;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.sap.cloud.account.TenantContext;
import com.sap.core.connectivity.api.configuration.ConnectivityConfiguration;
import com.sap.core.connectivity.api.configuration.DestinationConfiguration;
/**
 * Servlet class making http calls to specified http destinations.
 * Destinations are used in the following example connectivity scenarios:<br>
 * - Connecting to an outbound Internet resource using HTTP destinations<br>
 * - Connecting to an on-premise backend using on premise HTTP
 destinations,<br>
 * where the destinations have no authentication.<br>
 */
```

```

public class ConnectivityServlet extends HttpServlet {
    @Resource
    private TenantContext tenantContext;

    private static final long serialVersionUID = 1L;
    private static final int COPY_CONTENT_BUFFER_SIZE = 1024;
    private static final Logger LOGGER =
        LoggerFactory.getLogger(ConnectivityServlet.class);

    private static final String ON_PREMISE_PROXY = "OnPremise";

    /** {@inheritDoc} */
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpURLConnection urlConnection = null;
        String destinationName = request.getParameter("destname");

        // The default request to the Servlet will use outbound-internet-
destination
        if (destinationName == null) {
            destinationName = "outbound-internet-destination";
        }

        try {
            // Look up the connectivity configuration API
            Context ctx = new InitialContext();
            ConnectivityConfiguration configuration
= (ConnectivityConfiguration) ctx.lookup("java:comp/env/
connectivityConfiguration");

            // Get destination configuration for "destinationName"
            DestinationConfiguration destConfiguration =
configuration.getConfiguration(destinationName);
            if (destConfiguration == null) {

response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                    String.format("Destination %s is not found. Hint:
Make sure to have the destination configured.", destinationName));
                    return;
            }

            // Get the destination URL
            String value = destConfiguration.getProperty("URL");
            URL url = new URL(value);

            String proxyType = destConfiguration.getProperty("ProxyType");
            Proxy proxy = getProxy(proxyType);

            urlConnection = (HttpURLConnection) url.openConnection(proxy);

            // Insert the required header in the request for on-premise
destinations
            injectHeader(urlConnection, proxyType);

            // Copy content from the incoming response to the outgoing
response
            InputStream instream = urlConnection.getInputStream();
            OutputStream outstream = response.getOutputStream();
            copyStream(instream, outstream);
        } catch (Exception e) {
            // Connectivity operation failed
            String errorMessage = "Connectivity operation failed with reason:
"
                + e.getMessage()
                + ". See "
                + "logs for details. Hint: Make sure to have an HTTP
proxy configured in your "

```

```

        + "local environment in case your environment uses "
        + "an HTTP proxy for the outbound Internet "
        + "communication.";
    LOGGER.error("Connectivity operation failed", e);
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
        errorMessage);
    }
}

private Proxy getProxy(String proxyType) {
    String proxyHost = null;
    int proxyPort;
    if (ON_PREMISE_PROXY.equals(proxyType)) {
        // Get proxy for on-premise destinations
        proxyHost = System.getenv("HC_OP_HTTP_PROXY_HOST");
        proxyPort =
Integer.parseInt(System.getenv("HC_OP_HTTP_PROXY_PORT"));
    } else {
        // Get proxy for internet destinations
        proxyHost = System.getProperty("http.proxyHost");
        proxyPort =
Integer.parseInt(System.getProperty("http.proxyPort"));
    }
    return new Proxy(Proxy.Type.HTTP, new InetSocketAddress(proxyHost,
proxyPort));
}

private void injectHeader(HttpURLConnection urlConnection, String
proxyType) {
    if (ON_PREMISE_PROXY.equals(proxyType)) {
        // Insert header for on-premise connectivity with the consumer
subaccount name
        urlConnection.setRequestProperty("SAP-Connectivity-
ConsumerAccount", tenantContext.getAccountName());
    }
}

private void copyStream(InputStream inStream, OutputStream outStream)
throws IOException {
    byte[] buffer = new byte[COPY_CONTENT_BUFFER_SIZE];
    int len;
    while ((len = inStream.read(buffer)) != -1) {
        outStream.write(buffer, 0, len);
    }
}
}

```

Note

The given servlet can be run with different destination scenarios, for which user should specify the destination name as a requested parameter in the calling URL. In the case of on-premise connection to a backend system, the destination names should be **backend-no-auth-destination**. That is, it will be accessed at: `<application_URL>/?destname=backend-no-auth-destination`

Note

When accessing a destination with a specific authentication type, use [AuthenticationHeaderProvider API \[page 142\]](#) to get authentication headers and then inject them in all requests to this destination.

9. Save the Java editor and make sure the project compiles without errors.

4. Deploy the Application

Note

We only recommend but not obligate that you create the destination before starting the application.

1. To deploy your Web application locally or on the cloud, follow the steps described in the respective pages:
2. Once the application is successfully deployed locally or on the cloud, the application issues an exception saying that the **backend-no-auth-destination** destination has not been specified yet:

```
HTTP Status 500 - Destination backend-no-auth-destination is not found. Hint:
Make sure to have the destination configured.
```

3. As a next step, you need to configure **backend-no-auth-destination**.


For more information, see [ConnectivityConfiguration API \[page 140\]](#).

5. Configure the Destination in the Cloud

To configure the destination in SAP BTP, you need to use the virtual host name (**virtualpingbackend**) and port (**1234**) specified in one of the previous steps on the Cloud Connector's [Access Control](#) tab page.

Note

- On-premise destinations support HTTP connections only.
- The connection from an application to the Cloud Connector (through the tunnel) is encrypted on TLS level. Also, you can choose between using HTTP or HTTPS to hop from the Cloud Connector to the back end.

1. In the Eclipse IDE, open the [Servers](#) view and double-click `<application>.<subaccount>` to open the cloud server editor.
2. Open the [Connectivity](#) tab page.
3. In the [All Destinations](#) section, choose  to create a new destination with the name **backend-no-auth-destination**.
4. Use the following configuration:

```
Name=backend-no-auth-destination
Type=HTTP
URL=http://virtualpingbackend:1234/BackendAppHttpNoAuth/noauth
Authentication=NoAuthentication
ProxyType=OnPremise
CloudConnectorVersion=2
```

All Destinations
Define and edit connectivity destinations.

backend-no-auth-des

Basic Properties
Set the basic properties of the selected HTTP destination.

URL:

Authentication:

Proxy Type:

Additional Properties
Set additional properties of the selected HTTP destination.

Name	Value
CloudConnectorVersion	2

Overview Connectivity Loggers

5. Save the destination.
6. The [Connectivity](#) editor automatically saves the configuration in the cloud.
7. Call the URL that references the cloud application again in the internal Web browser. The application should now return the ping response.

Related Information

[JavaDoc ConnectivityConfiguration](#)

[JavaDoc DestinationConfiguration](#)

[JavaDoc AuthenticationHeaderProvider](#)

[AuthenticationHeaderProvider API \[page 142\]](#)

1.4.1.2.3.5 Set Up an Application as a Sample Backend System


Overview

This section describes how you set up a simple ping Web application that is used as a backend system.

Prerequisites

You have downloaded SAP BTP SDK on your local file system.

Procedure

1. Set up a servlet container such as [Tomcat](#) .
2. Add a user and role for basic authentication by adding the following lines to the `tomcat-users.xml` file in directory `<TOMCAT_HOME>/conf` file:

```
<role rolename="pingrole"/>
<user name="pinguser" password="pingpassword" roles="pingrole" />
```

3. From the SDK location, go to `/samples/connectivity/onpremise`, copy files `BackendAppHttpNoAuth.war` and `BackendAppHttpBasicAuth.war` and paste them into the `<TOMCAT_HOME>/webapps` directory.
4. Start Tomcat and access the on-premise applications at the URLs below. Use `pinguser` / `pingpassword` as the credentials.
 - `http://localhost:8080/BackendAppHttpNoAuth/noauth`
 - `http://localhost:8080/BackendAppHttpBasicAuth/basic`

Note

In case you use SDK with version equal to or lower than, respectively, **1.44.0.1** (Java Web) and **2.24.13** (Java EE 6 Web Profile), you should find the WAR files in directory `<SDK_location>/tools/samples/connectivity/onpremise`, under the names **PingAppHttpNoAuth.war** and **PingAppHttpBasicAuth.war**. Also, you should access the applications at the relevant URLs:

- `http://localhost:8080/PingAppHttpNoAuth/pingnoauth`
- `http://localhost:8080/PingAppHttpBasicAuth/pingbasic`

Related Information



[Consume Backend Systems \(Java Web or Java EE 6 Web Profile\) \[page 171\]](#)

1.4.1.3 Invoking ABAP Function Modules via RFC

Call a remote-enabled function module in an on-premise ABAP server from your Neo application, using the RFC protocol.

Find the tasks and prerequisites that are required to consume an on-premise ABAP function module via RFC, using the Java Connector (JCo) API as a built-in feature of SAP BTP.

Tasks

Task Type	Task
	Prerequisites [page 192]
Operator	
	About JCo [page 192]
	Installation Prerequisites for JCo Applications [page 193]
Operator and/or Developer	Consume Connectivity via RFC [page 193]
	Restrictions [page 194]

Prerequisites



Before you can use RFC communication for an SAP BTP application, you must configure:

- A destination on SAP BTP to use RFC.
For more information, see [Configure Destinations from the Console Client \[page 55\]](#) and [RFC Destinations \[page 114\]](#).
- RFC connectivity between a backend system and the application. To do this, you must install the [Cloud Connector \[page 232\]](#) in your internal network and configure it to expose a remote-enabled function module in an on-premise ABAP system.
For more information, see [Initial Configuration \(RFC\) \[page 287\]](#) and [Configure Access Control \(RFC\) \[page 350\]](#).

Back to [Tasks \[page 191\]](#)

About JCo



To learn in detail about the SAP JCo API, see the JCo 3.0 documentation on [SAP Support Portal](#).

Note

Some sections of this documentation are not applicable to SAP BTP:

- **Architecture:** CPIC is only used in the last mile from your Cloud Connector to the back end. From SAP BTP to the Cloud Connector, TLS-protected communication is used.

- **Installation:** SAP BTP runtimes already include all required artifacts.
- **Customizing and Integration:** On SAP BTP, the integration is already done by the runtime. You can concentrate on your business application logic.
- **Server Programming:** The programming model of JCo on SAP BTP does not include server-side RFC communication.
- **IDoc Support for External Java Applications:** Currently, there is no IDocLibrary for JCo available on SAP BTP

Back to [Tasks \[page 191\]](#)

Installation Prerequisites for JCo Applications



To develop a JCo application:

- your SDK version must be **1.29.18** (*SDK Java Web*), or **2.11.6** (*SDK for Java EE 6 Web Profile*).
- your SDK local runtime must be hosted by a 64-bit JVM. SDKs of Tomcat 7, Tomcat 8, and TomEE 7 runtime support JCo from the very beginning.
- on Windows platforms, you must install **Microsoft Visual Studio C++ 2013 runtime libraries (vcredist_x64.exe)**. To download this package, go to <https://www.microsoft.com/en-us/download/details.aspx?id=40784> .

Back to [Tasks \[page 191\]](#)

Consume Connectivity via RFC



You can call a service from a fenced customer network using a simple application which consumes a simple on-premise, remote-enabled function module.

Invoking function modules via RFC is enabled by a JCo API that is comparable to the one available in SAP NetWeaver Application Server Java (since version 7.10), and in JCo standalone since version 3.0. If you are an experienced JCo developer, you can easily develop a Web application using JCo: you simply consume the APIs like you do in other Java environments. Restrictions that apply in the cloud environment are mentioned in the **Restrictions** section below.

Find a sample Web application in [Invoke ABAP Function Modules in On-Premise ABAP Systems \[page 195\]](#).

Back to [Tasks \[page 191\]](#)

Restrictions



- **JCoServer** functionality cannot be used within SAP BTP.
- **Environment embedding**, such as offered by JCo standalone 3.1, is not possible. This is, however, similar to SAP NetWeaver AS Java.
- By default, a **stateful sequence of function module invocations** must be done in a single HTTP request/response cycle.

To make a stateful sequence work across several request/response cycles, you must set the parameter `advancedJCoUsage`. In the cloud cockpit, navigate to your subaccount, and from the left-side subaccount menu choose ► [Applications](#) ► [Java Applications](#) ► [Deploy Application](#) . In the pop-up window, enter **-DadvancedJCoUsage=true** in the field `<JVM Arguments>`. Alternatively, you can specify the argument when deploying with the Neo SDK tools.

Note

To add the parameter to an existing application, select the application and choose [Update](#). When you are done, you must restart the application.

The minimal runtime versions for supporting this capability are listed below:

Runtime/SDK	Minimal Version
Java Web	1.171.11.1
Java Web Tomcat 7	2.118.20.1
Java Web Tomcat 8	3.73.18.1
Java EE 6 Web Profile	2.153.9.1
Java EE 7 Web Profile TomEE 7	1.42.19.1

- **Logon authentication** only supports user/password credentials (basic authentication) and principal propagation. See [Create RFC Destinations \[page 81\]](#) and [User Logon Properties \[page 114\]](#).
- **Provider/subscription model** for applications is only fully supported in newer runtime versions. If you still want to use it in older ones, you need to make sure that destinations are named differently in all accounts. Minimal runtime versions for full support are listed below:

Runtime/SDK	Minimal Version
Java Web	1.142.0
Java Web Tomcat 7	2.89.13
Java Web Tomcat 8	3.43.12

Runtime/SDK	Minimal Version
Java EE 6 Web Profile	2.124.0
Java EE 7 Web Profile TomEE 7	1.13.12

- The supported set of **configuration properties** is restricted. For details, see [RFC Destinations \[page 114\]](#).

Back to [Tasks \[page 191\]](#)

Related Information

[Java Connector API \[page 142\]](#)

1.4.1.3.1 Invoke ABAP Function Modules in On-Premise ABAP Systems

Context

This example shows how a sample Web application invokes a function module in an on-premise ABAP system via RFC by using theConnectivity service.

The procedure contains the following sections:

- Presenting the user roles
- Defining the installation prerequisites
- Developing a sample Web application that uses theConnectivity service to consume the simple function module `STFC_CONNECTION`.

Connectivity User Roles

Different user roles are involved in the on-demand to on-premise connectivity end-to-end scenario. The particular steps for the relevant roles are described below:

IT Administrator

This role sets up and configures the Cloud Connector. Scenario steps:

1. Downloads the Cloud Connector from <https://tools.hana.ondemand.com/#cloud>
2. Installs the Cloud Connector.
3. Establishes an TLS tunnel from the connector to an SAP BTP subaccount.
4. Configures the exposed backend systems and resources.

Application Developer

This role develops Web applications using destinations. Scenario steps:

1. Installs the Eclipse IDE, SAP BTP Tools for Java, and SDK.
2. Develops a Java EE application using the destination API.
3. Configures connectivity destinations as resources in the *web.xml* file.
4. Configures connectivity destinations via the SAP BTP server adapter in Eclipse IDE.
5. Deploys the Java EE application locally and on the cloud.

Subaccount Operator


This role deploys Web applications, configures their destinations, and conducts tests. Scenario steps:

1. Obtains a ready Java EE application WAR file.
2. Deploys the Java EE application in an SAP BTP subaccount.
3. Uploads the connectivity destination configuration via the console client.
4. Tests the Java EE application on a local server and deploys it again to a SAP BTP subaccount.

Installation Prerequisites

- You have downloaded and set up your Eclipse IDE and SAP BTP Tools for Java.
- You have downloaded the SDK. Its version needs to be at least **1.29.18** (*SDK for Java Web*), **2.11.6** (*SDK for Java EE 6 Web Profile*), or **2.9.1** (*SDK for Java Web Tomcat 7*), respectively.
- Your local runtime needs to be hosted by a 64-bit JVM. On Windows platforms, you need to install *Microsoft Visual C++ 2010 Redistributable Package (x64)*.
- You have downloaded and configured your Cloud Connector. Its version needs to be at least **1.3.0**.

To download the SAP tools, go to <https://tools.hana.ondemand.com/#cloud>.

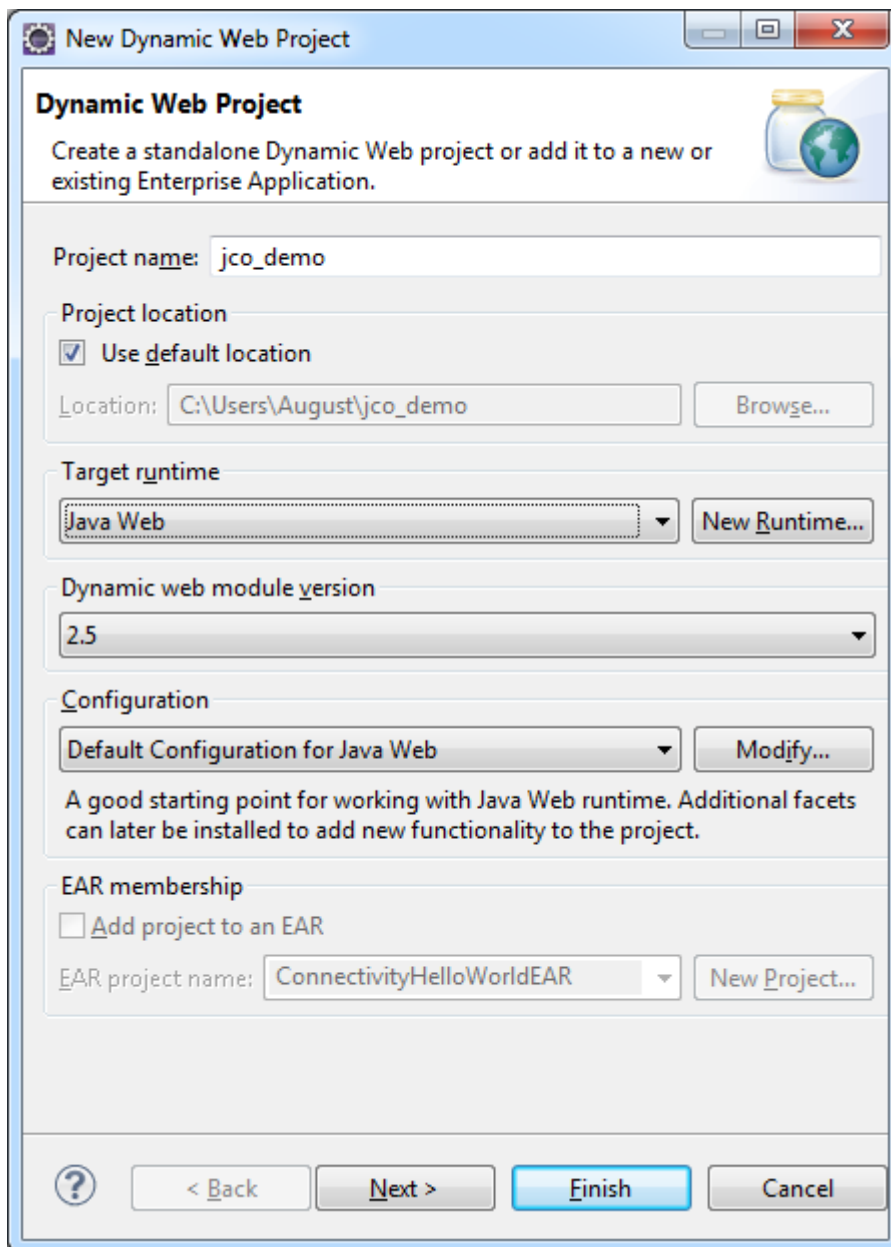
To download the Microsoft Visual C++ package, go to <http://www.microsoft.com/en-us/download/details.aspx?id=14632> .

To read the installation documentation, go to [Setting Up the Development Environment](#) and [Installation \[page 238\]](#).

Creating a Dynamic Web Project

Procedure

1. In the Eclipse IDE, open the *Java EE* perspective.
2. From the Eclipse main menu, choose **New** > *Dynamic Web Project*.
3. In the *Project name* field, enter `jco_demo`.
4. In the *Target Runtime* pane, select the runtime you want to use to deploy the HelloWorld application. In this example, we choose **Java Web**.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to complete the creation of your project.



Creating a Sample Servlet

Procedure

1. From the `jco_demo` context menu, choose **New** > **Servlet**.
2. Enter `com.sap.demo.jco` as the *Java package* and `ConnectivityRFCExample` as the *Class name* and choose **Next**.
3. Choose **Finish** so that the `ConnectivityRFCExample.java` servlet is created and opened in the Java editor.
4. Replace the entire servlet class to make use of the JCo API. The JCo API is visible by default for cloud applications and must not be added explicitly to the application class path.

```
package com.sap.demo.jco;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.sap.conn.jco.AbapException;
import com.sap.conn.jco.JCoDestination;
import com.sap.conn.jco.JCoDestinationManager;
import com.sap.conn.jco.JCoException;
import com.sap.conn.jco.JCoFunction;
import com.sap.conn.jco.JCoParameterList;
import com.sap.conn.jco.JCoRepository;
/**
 * Sample application that uses the Connectivity
 *                               service. In particular,
 * it makes use of the capability to invoke a function module in an ABAP
system
 * via RFC
 *
 * Note: The JCo APIs are available under <code>com.sap.conn.jco</code>.
 */
public class ConnectivityRFCExample extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    public ConnectivityRFCExample()
    {
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException
    {
        PrintWriter responseWriter = response.getWriter();
        try
        {
            // access the RFC Destination "JCoDemoSystem"
            JCoDestination
destination=JCoDestinationManager.getDestination("JCoDemoSystem");
            // make an invocation of STFC_CONNECTION in the backend;
            JCoRepository repo=destination.getRepository();
            JCoFunction stfcConnection=repo.getFunction("STFC_CONNECTION");
            JCoParameterList imports=stfcConnection.getImportParameterList();
            imports.setValue("REQUTEXT", "SAP Connectivity service runs with
JCo");
            stfcConnection.execute(destination);
            JCoParameterList exports=stfcConnection.getExportParameterList();
```

```

        String echotext=exports.getString("ECHOTEXT");
        String resptext=exports.getString("RESPTEXT");
        response.addHeader("Content-type", "text/html");
        responseWriter.println("<html><body>");
        responseWriter.println("<h1>Executed STFC_CONNECTION in system
JCoDemoSystem</h1>");
        responseWriter.println("<p>Export parameter ECHOTEXT of
STFC_CONNECTION:<br>");
        responseWriter.println(echotext);
        responseWriter.println("<p>Export parameter RESPTEXT of
STFC_CONNECTION:<br>");
        responseWriter.println(resptext);
        responseWriter.println("</body></html>");
    }
    catch (AbapException ae)
    {
        //just for completeness: As this function module does not have an
exception
        //in its signature, this exception cannot occur. However, you
should always
        //take care of AbapExceptions
    }
    catch (JCoException e)
    {
        response.addHeader("Content-type", "text/html");
        responseWriter.println("<html><body>");
        responseWriter.println("<h1>Exception occurred while executing
STFC_CONNECTION in system JCoDemoSystem</h1>");
        responseWriter.println("<pre>");
        e.printStackTrace(responseWriter);
        responseWriter.println("</pre>");
        responseWriter.println("</body></html>");
    }
}
}
}

```

5. Save the Java editor and make sure that the project compiles without errors.

Deploying the Application

Procedure

1. To deploy your Web application locally or on the cloud, see the following two procedures, respectively:
 -
 -
2. Once the application is successfully deployed on the cloud and you execute it, the application throws an exception. This exception says that the *JCoDemoSystem* destination has not been specified yet:

```

Exception occurred while executing STFC_CONNECTION in system JCoDemoSystem
com.sap.conn.jco.JCoException: (106) JCO_ERROR_RESOURCE: Destination
JCoDemoSystem does not exist
    at
com.sap.conn.jco.rt.DefaultDestinationManager.update(DefaultDestinationManager
.java:223)
    at
com.sap.conn.jco.rt.DefaultDestinationManager.searchDestination(DefaultDestina
tionManager.java:377)

```

```

        at
com.sap.conn.jco.rt.DefaultDestinationManager.getDestinationInstance(DefaultDe
stinationManager.java:96)
        at
com.sap.conn.jco.JCoDestinationManager.getDestination(JCoDestinationManager.ja
va:52)
        at
com.sap.demo.jco.ConnectivityRFCExample.doGet(ConnectivityRFCExample.java:47)
        .... (cut rest of the call stack)

```

3. As a next step, you need to configure the *JCoDemoSystem* destination.

Configuring the RFC Destination on the Cloud

To configure the destination on SAP BTP, you need to use a virtual application server host name (**abapserver.hana.cloud**) and a virtual system number (**42**) that you will expose later in the Cloud Connector. Alternatively, you could use a load balancing configuration with a message server host and a system ID.

Procedure

1. Create a properties file with the following settings:

```

Name=JCoDemoSystem
Type=RFC
jco.client.ashost=abapserver.hana.cloud
jco.client.cloud_connector_version=2
jco.client.sysnr=42
jco.client.user=DEMOUSER
jco.client.passwd=<password>
jco.client.client=000
jco.client.lang=EN
jco.destination.pool_capacity=5

```

2. Upload this file to your Web application in SAP BTP. For more information, see [Configure Destinations from the Console Client \[page 55\]](#).
3. Call the URL that references the cloud application again in the Web browser. The application should now return a different exception:

```

Exception occurred while executing STFC_CONNECTION in system JCoDemoSystem
com.sap.conn.jco.JCoException: (102) JCO_ERROR_COMMUNICATION: Opening
connection to backend failed: Opening connection denied
        at
com.sap.conn.jco.rt.MiddlewareJavaRfc.generateJCoException(MiddlewareJavaRfc.j
ava:632)
        at
com.sap.conn.jco.rt.MiddlewareJavaRfc$JavaRfcClient.connect(MiddlewareJavaRfc.
java:1307)
            at com.sap.conn.jco.rt.ClientConnection.connect(ClientConnection.java:726)
            at com.sap.conn.jco.rt.PoolingFactory.init(PoolingFactory.java:107)
            at
com.sap.conn.jco.rt.ConnectionManager.createFactory(ConnectionManager.java:316
)
            at
com.sap.conn.jco.rt.DefaultConnectionManager.createFactory(DefaultConnectionMa
nager.java:46)

```

```

at
com.sap.conn.jco.rt.ConnectionManager.getFactory(ConnectionManager.java:290)
at
com.sap.conn.jco.rt.ConnectionManager.getClient(ConnectionManager.java:83)
at com.sap.conn.jco.rt.Context.getConnection(Context.java:216)
at com.sap.conn.jco.rt.RfcDestination.execute(RfcDestination.java:1306)
at com.sap.conn.jco.rt.RfcDestination.execute(RfcDestination.java:1278)
at com.sap.conn.jco.rt.AbapFunction.execute(AbapFunction.java:295)
at
com.sap.demo.jco.ConnectivityRFCEXample.doGet(ConnectivityRFCEXample.java:55)
..... (cut rest of the call stack)

```

4. This means the Cloud Connector denied opening a connection to this system. As a next step, you need to configure the system in your installed Cloud Connector.

Configuring the System Mapping in the Cloud Connector

This is required since the Cloud Connector only allows access to white-listed backend systems. To do this, follow the steps below:

Procedure

1. Optional: In the Cloud Connector administration UI, you can check under [Audits](#) whether access has been denied:

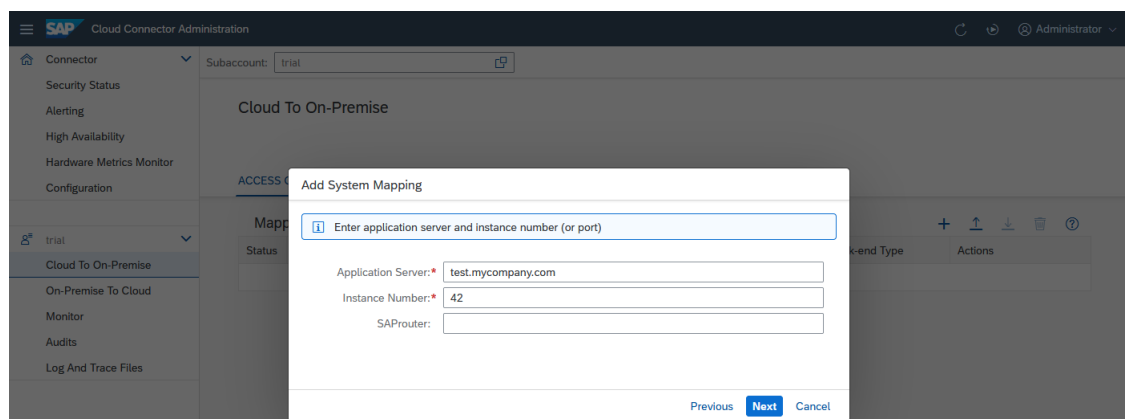
```

Denying access for user DEMOUSER to system abapserver.hana.cloud:sapgw42
[connectionId=-1547299395]

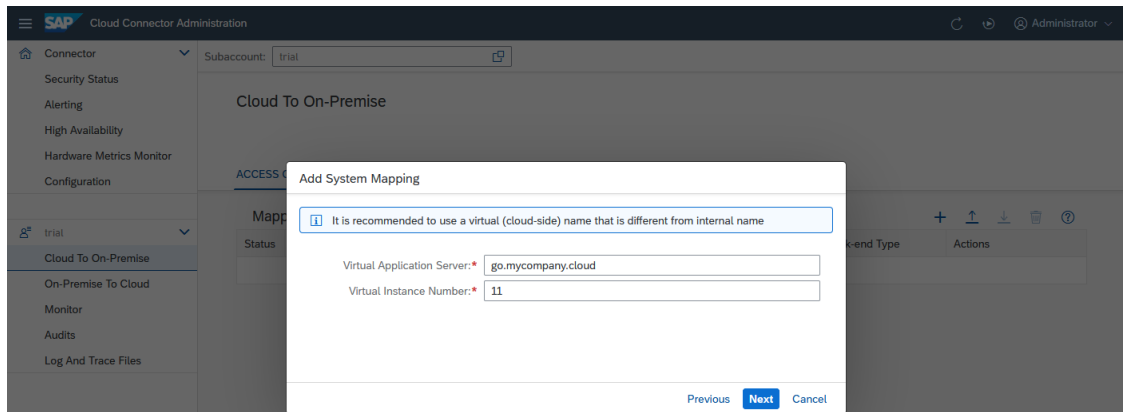
```

2. In the Cloud Connector administration UI and choose [Cloud To On-Premise](#) from your [Subaccount](#) menu, tab [Access Control](#).
3. In section [Mapping Virtual To Internal System](#) choose [Add](#) to define a new system.
 1. For [Backend Type](#), select **ABAP System** and choose [Next](#).
 2. For [Protocol](#), select **RFC** and choose [Next](#).
 3. Choose option [Without load balancing](#).
 4. Enter application server and instance number. The [Application Server](#) entry must be the physical host name of the machine on which the ABAP application server is running. Choose [Next](#).

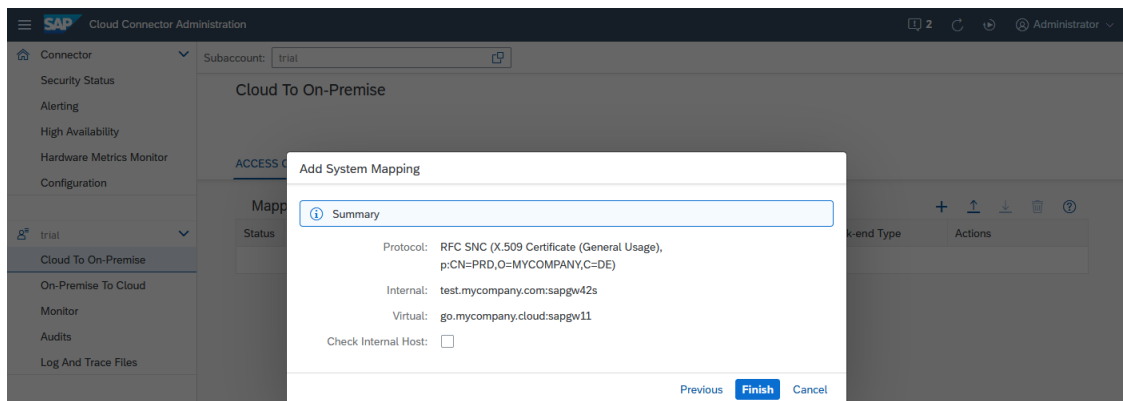
Example:



5. Enter server and instance number for virtual mapping. **Example:**



6. Summary (example):



4. Call the URL that references the cloud application again in the Web browser. The application should now throw a different exception:

```
com.sap.conn.jco.JCoException: (102) JCO_ERROR_COMMUNICATION: Access denied
for STFC_CONNECTION
    at
com.sap.conn.jco.rt.MiddlewareJavaRfc.generateJCoException(MiddlewareJavaRfc.j
ava:632)
    at
com.sap.conn.jco.rt.MiddlewareJavaRfc$JavaRfcClient.execute(MiddlewareJavaRfc.
java:1764)
    at
com.sap.conn.jco.rt.ClientConnection.execute(ClientConnection.java:1110)
    at com.sap.conn.jco.rt.ClientConnection.execute(ClientConnection.java:943)
    at com.sap.conn.jco.rt.RfcDestination.execute(RfcDestination.java:1307)
    at com.sap.conn.jco.rt.RfcDestination.execute(RfcDestination.java:1278)
    at com.sap.conn.jco.rt.AbapFunction.execute(AbapFunction.java:295)
    at
com.sap.demo.jco.ConnectivityRFCExample.doGet(ConnectivityRFCExample.java:55)
    .... (cut rest of the call stack)
```

5. This means the Cloud Connector denied invoking STFC_CONNECTION in this system. As a final step, you need to provide access to this function module in your installed Cloud Connector.

Configuring the Function Module in the Cloud Connector

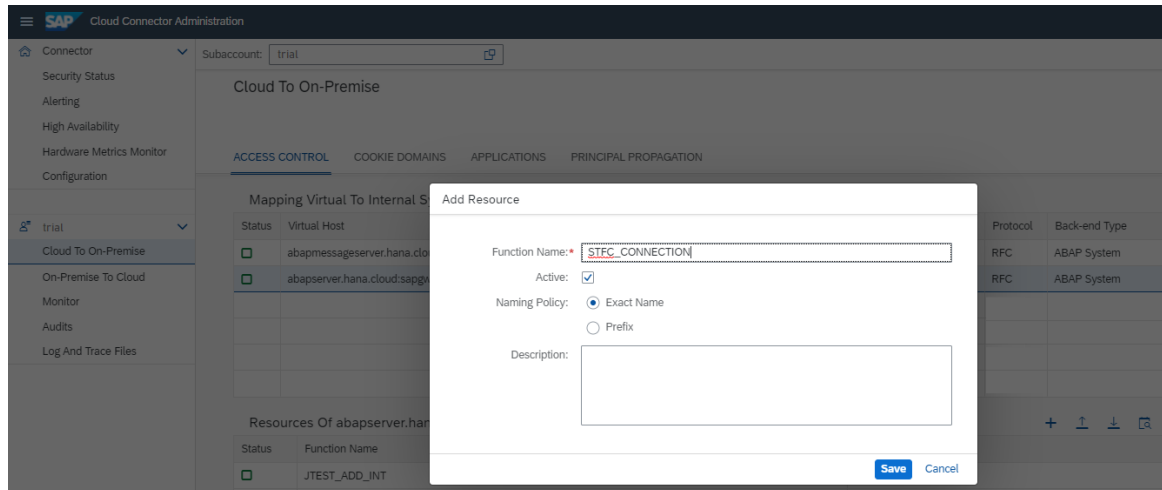
This is required since the Cloud Connector only allows access to white-listed resources (which are defined on the basis of function module names with RFC). To do this, follow the steps below:

Procedure

1. Optional: In the Cloud Connector administration UI, you can check under [Audits](#) whether access has been denied:

```
Denying access for user DEMOUSER to resource STFC_CONNECTION on system
abapserver.hana.cloud:sapgw42 [connectionId=609399452]
```

2. In the Cloud Connector administration UI, choose [Cloud To On-Premise](#) from your [Subaccount](#) menu, and go to the [Access Control](#) tab.
3. For the specified internal system referring to [abapserver.hana.cloud](#), add a new resource. Select the system in the table.
4. Add a new function name under the list of exposed resources. In section [Resources Accessible On abapserver.hana.cloud:sapgw42](#), choose the [Add](#) button and specify STFC_CONNECTION as the accessible resource as shown in the screenshot below. Make sure that you have selected the [Exact Name](#) option to only expose this single function module.



5. Call the URL that references the cloud application again in the Web browser. The application should now return with a message showing the export parameters of the function module after a successful invocation.

1.4.1.4 Using LDAP

Find an example how to use an LDAP destination within your cloud application.

To learn more about configuring LDAP destinations, see [LDAP Destinations \[page 126\]](#).

Example: LDAP Destination

Sample Code

```
package com.sap.cloud.example.ldap;

import java.io.IOException;
import java.util.Properties;

import javax.annotation.Resource;
import javax.naming.NamingEnumeration;
import javax.naming.NamingException;
import javax.naming.directory.DirContext;
import javax.naming.directory.InitialDirContext;
import javax.naming.directory.SearchControls;
import javax.naming.directory.SearchResult;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.sap.core.connectivity.api.configuration.ConnectivityConfiguration;
import com.sap.core.connectivity.api.configuration.DestinationConfiguration;

/**
 * Servlet that obtain LDAP destination, connect to the specified LDAP server
 * and search for users.
 */
@WebServlet("/*")
public class LdapExample extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final String DESTINATION_NAME = "example-ldap-destination";

    private static final String LDAP_PATH_TO_USERS =
"ou=users,dc=examplecompany,dc=com";
    private static final String LDAP_FILTER_MATCHING_USERS =
"(objectClass=person)";

    @Resource(name = "ConnectivityConfiguration")
    private static ConnectivityConfiguration connectivityConfiguration;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        DestinationConfiguration destination =
connectivityConfiguration.getConfiguration(DESTINATION_NAME);
        Properties properties = new Properties();
        properties.putAll(destination.getAllProperties());

        try {
            DirContext context = new InitialDirContext(properties);
            SearchControls controls = new SearchControls();
            controls.setSearchScope(SearchControls.SUBTREE_SCOPE);

            NamingEnumeration<SearchResult> result =
context.search(LDAP_PATH_TO_USERS, LDAP_FILTER_MATCHING_USERS, controls);
            response.getWriter().append("Found users:<br/><br/>");
            while (result.hasMore()) {

response.getWriter().append(result.next().toString()).append("<br/><br/>");
            }
        } catch (NamingException e) {
            throw new ServletException("Could not search LDAP for users", e);
        }
    }
}
```


1.4.1.5 Using the TCP Protocol for Cloud Applications

Access on-premise systems from a Neo application via TCP-based protocols, using a SOCKS5 Proxy.

Content

[Concept \[page 205\]](#)


[Restrictions \[page 206\]](#)

[Example \[page 206\]](#)

[Interfaces \[page 207\]](#)

[Troubleshooting \[page 207\]](#)

Concept

SAP BTP Connectivity provides a SOCKS5 proxy that you can use to access on-premise systems via TCP-based protocols. SOCKS5 is the industry standard for proxying TCP-based traffic (for more information, see IETF [RFC 1928](#) ).

The proxy server is started by default on all application machines. So you can access it on `localhost` and port `20004`.

The use of SOCK5 *Username/Password Authentication* is mandatory as it is required to provide routing information to the proxy, see [RFC 1929](#) .

In this scenario, it is used to find the correct Cloud Connector to which the data will be routed. Therefore, the pattern used for username is `1.<subaccount>.<locationId>`, where `subaccount` is a mandatory parameter, whereas `locationId` is optional.

Note

Both values should be base64-encoded.

The Cloud Connector location ID identifies Cloud Connector instances that are deployed in various locations of a customer's premises and connected to the same subaccount. Since the location ID is an optional property, you should include it in the request only if it has already been configured in the Cloud Connector. For more information, see [Set up Connection Parameters and HTTPS Proxy \[page 281\]](#) (Step 4).

The `password` part of the authentication scheme is left as an empty string in this scenario.

Back to [Content \[page 205\]](#)

Restrictions

- You can use the provided SOCKS5 proxy server only to connect to on-premise systems. You cannot use it as general-purpose SOCKS5 proxy.
- Proxying UDP traffic is not supported.

Back to [Content \[page 205\]](#)

Example

The following code snippet shows how to provide the proxy authentication values :

Sample Code

```
import java.net.Authenticator;
import org.apache.commons.codec.binary.Base64; // Or any other Base64 encoder

private void setSOCKS5ProxyAuthentication(String subaccount, String
locationId){
    final String encodedSubaccount = new
String(Base64.encodeBase64(subaccount.getBytes()));
    final String encodedLocationId = new
String(Base64.encodeBase64(locationId.getBytes()));

    Authenticator.setDefault(new Authenticator() {
        @Override
        protected java.net.PasswordAuthentication getPasswordAuthentication()
        {
            return new java.net.PasswordAuthentication("1." +
encodedSubaccount + "." + encodedLocationId , new char[]{});
        }
    });
}
```

In this code snippet you can see how to set up the SOCKS proxy and how to use it to create an HTTP connection:

Sample Code

```
import java.net.SocketAddress;
import java.net.InetSocketAddress;
import java.net.Proxy;
import java.net.URL;
import java.net.HttpURLConnection;

SocketAddress addr = new InetSocketAddress("localhost", 20004);
Proxy proxy = new Proxy(Proxy.Type.SOCKS, addr);
setSOCKS5ProxyAuthentication(subaccount, locationId); // where subaccount
is the current subaccount and locationId is the Location ID of the SCC (or
empty string if locationId is not set)
URL url = new URL("http://virtualhost:1234/");
```

```
HttpURLConnection connection = (HttpURLConnection)
url.openConnection(proxy);
```

Back to [Content](#) [page 205]

Interfaces


You can access a subaccount associated with the current execution thread using the `TenantContext` API.

See also:

- [Interface TenantContext](#)
- [Interface TenantContext: getTenant\(\)](#)
- [Interface Tenant: getAccount\(\)](#)
- [Interface Account: getId\(\)](#)

Back to [Content](#) [page 205]

Troubleshooting

If the handshake with the SOCKS5 proxy server fails, a SOCKS5 protocol error is returned, see IETF [RFC 1928](#) . The table below shows the most common errors and their root cause in the scenario you use:

SOCKS5 Error Code	Technical Description	Client-Side Error Description	Scenario Error
0x00	SUCCESS		Success
0x01	FAILURE		Connection closed by back-end or general scenario failure.
0x02	FORBIDDEN	Connection not allowed by ruleset	No matching host mapping found in Cloud Connector access control settings, see Configure Access Control (TCP) [page 359].

SOCKS5 Error Code	Technical Description	Client-Side Error Description	Scenario Error
0x03	NETWORK_UNREACHABLE		The Cloud Connector is not connected to the subaccount and the Cloud Connector Location ID that is used by the cloud application can't be identified. See Connect and Disconnect a Cloud Subaccount [page 544] and Managing Subaccounts [page 291] , section <i>Procedure</i> .
0x04	HOST_UNREACHABLE		Cannot open connection to the backend, that is, the host is unreachable.
0x05	CONNECTION_REFUSED		<i>Not used</i>
0x06	TTL_EXPIRED		<i>Not used</i>
0x07	COMMAND_UNSUPPORTED		Only the SOCKS5 CONNECT command is supported.
0x08	ADDRESS_UNSUPPORTED		Only the SOCKS5 DOMAIN and IPv4 commands are supported.

Back to [Content \[page 205\]](#)

Related Information

[Configure Access Control \(TCP\) \[page 359\]](#)

1.4.1.6 Sending and Fetching E-Mail

E-mail connectivity lets you send messages from your Web applications using e-mail providers that are accessible on the Internet, as well as retrieve e-mails from the mailbox of your e-mail account.

Note

SAP does not act as e-mail provider. To use this service, please cooperate with an external e-mail provider of your choice.

To send and fetch e-mail, you must do the following:

- Obtain a mail session resource using resource injection or, alternatively, using a JNDI lookup.

- Configure the mail session resource by specifying the protocol settings of your mail server as a mail destination configuration. SMTP+STARTTLS (Port 587) and SMTPS (Port 465) are supported for sending e-mail, and POP3 and IMAP for retrieving messages from a mailbox account.
- In your Web application, use the JavaMail API (`javax.mail`) to create and send a `MimeMessage` object or retrieve e-mails from a message store.

Related Information

[Mail Destinations \[page 128\]](#)

[JavaMail API \[page 209\]](#)

[Enable the Debugging Feature \[page 211\]](#)

[Example: Send E-Mails \[page 212\]](#)

1.4.1.6.1 JavaMail API

In your Web application, you use the JavaMail API (`javax.mail`) to create and send a `MimeMessage` object or retrieve e-mails from a message store.

Mail Session

You can obtain a mail session resource using resource injection or a JNDI lookup. The properties of the mail session are specified by a mail destination configuration. So that the resource is linked to this configuration, the names of the destination configuration and mail session resource must be the same.

- Resource injection
You can directly inject the mail session resource using annotations as shown in the example below. You do not need to declare the JNDI resource reference in the `web.xml` deployment descriptor.

```
@Resource(name = "mail/Session")
private javax.mail.Session mailSession;
```

- JNDI lookup
To obtain a resource of type `javax.mail.Session`, you declare a JNDI resource reference in the `web.xml` deployment descriptor in the `webContent/WEB-INF` directory as shown below. Note that the recommended resource reference name is `Session` and the recommended subcontext is `mail` (`mail/Session`):

```
<resource-ref>
  <res-ref-name>mail/Session</res-ref-name>
  <res-type>javax.mail.Session</res-type>
</resource-ref>
```

An initial JNDI context can be obtained by creating a `javax.naming.InitialContext` object. You can then consume the resource by looking up the naming environment through the `InitialContext`, as follows:

```
InitialContext ctx = new InitialContext();
Session mailSession = (Session)ctx.lookup("java:comp/env/mail/Session");
```

Note that according to the Java EE Specification, the prefix `java:comp/env` should be added to the JNDI resource name (as specified in the `web.xml`) to form the lookup name.

Sending E-Mail

With the `javax.mail.Session` object you have retrieved, you can use the JavaMail API to create a `MimeMessage` object with its constituent parts (instances of `MimeMultipart` and `MimeBodyPart`). The message can then be sent using the `send` method from the `Transport` class:

```
Transport transport = mailSession.getTransport();
transport.connect();
MimeMessage mimeMessage = new MimeMessage(mailSession);
...
transport.sendMessage(mimeMessage, mimeMessage.getAllRecipients());
transport.close();
```

Fetching E-Mail

You can retrieve the e-mails from the inbox folder of your e-mail account using the `getFolder` method from the `Store` class as follows:

```
Store store = mailSession.getStore();
store.connect();
Folder folder = store.getFolder("INBOX");
folder.open(Folder.READ_ONLY);
Message[] messages = folder.getMessages();
...
folder.close(true);
store.close();
```

Fetches e-mail is not scanned for viruses. This means that e-mail retrieved from an e-mail provider using IMAP or POP3 could contain a virus that could potentially be distributed (for example, if e-mail is stored in the database or forwarded). Basic mitigation steps you could take include the following:

- Choose an e-mail provider that scans received e-mail for viruses
- Store e-mail in the document service repository before processing it. Make sure that the virus scanner provided by the document service is enabled.
- Generally don't resend e-mail that you have fetched

Related Information

[Connectivity and Destination APIs \[page 136\]](#)

1.4.1.6.2 Enable the Debugging Feature

In order to troubleshoot e-mail delivery and retrieval issues, it is useful to have debug information about the mail session established between your SAP BTP application and your e-mail provider.

Context

To include debug information in the standard trace log files written at runtime, you can use the JavaMail debugging feature and the `System.out` logger. The `System.out` logger is preconfigured with the log level INFO. You require at least INFO or a level with more detailed information.

Procedure

1. To enable the JavaMail debugging feature, add the `mail.debug` property to the mail destination configuration as shown below:

```
mail.debug=true
```

2. To check the log level for your application, log on to the cockpit.
3. In the content area, choose **Applications** > **Java Applications**.
4. In the application list, select your application to go to the overview.
5. In the content area, choose **Monitoring** > **Logging**.
6. In the **Default Trace** section in the **Log Files** panel, choose **Configure Loggers**.
In the **Logger Configuration** dialog box, all loggers used since the application was started are listed with the log levels that are currently applicable. Loggers are listed only if the relevant application code has been executed.
7. Enter `system.out` in the **Filter** field.
8. If necessary, change the log level for the `System.out` logger.

Note

You can check the log level of the `System.out` logger in a similar manner from the Eclipse IDE.

1.4.1.6.3 Example: Send E-Mails

This example shows how you can send an e-mail from a simple Web application using an e-mail provider that is accessible on the Internet.

❗ Note

SAP does not act as e-mail provider. To use this service, please cooperate with an external e-mail provider of your choice.

Steps	Sample Application
Prerequisites [page 212]	The application is also available as a sample in the SAP BTP SDK:
1. Create a Dynamic Web Project and Servlet [page 212]	
2. Extend the Servlet [page 213]	Sample name: <code>mail</code>
3. Test the Application Locally [page 215]	Location: <code><sdk>/samples</code> folder
4. Test the Application in the Cloud [page 215]	More information: Using Samples

Prerequisites

You have installed the SAP BTP Tools and created an SAP HANA Cloud server runtime environment as described in [Setting Up the Development Environment](#).



1. Create a Dynamic Web Project and Servlet

To develop applications for the SAP BTP, you require a dynamic Web project and servlet.

1. From the Eclipse main menu, choose **File > New > Dynamic Web Project**.
2. In the *Project name* field, enter `mail`.
3. In the *Target Runtime* pane, select the runtime you want to use to deploy the application. This example uses **Java Web**.
4. In the *Configuration* area, leave the default configuration and choose *Finish*.
5. To add a servlet to the project you have just created, select the `mail` node in the *Project Explorer* view.
6. From the Eclipse main menu, choose **File > New > Servlet**.
7. Enter the Java package `com.sap.cloud.sample.mail` and the class name `MailServlet`.
8. Choose *Finish* to generate the servlet.

2. Extend the Servlet

You add code to create a simple Web UI for composing and sending an e-mail message. The code includes the following methods:

- `doGet()`: Creates an HTML form for entering e-mail details.
 - `doPost()`: Uses the mail session resource to create and send a `MimeMessage` object. It confirms that an e-mail has been sent.
1. In the *Project Explorer* view, expand the `mail/Java Resources/src/com.sap.cloud.sample.mail` node.
 2. Select `MailServlet.java`, and from the context menu choose  *Open With* .
 3. In the opened editor, replace the entire servlet class with the following content:

```
package com.sap.cloud.sample.mail;
import java.io.IOException;
import java.io.PrintWriter;
import javax.annotation.Resource;
import javax.mail.Message.RecipientType;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
/**
 * Servlet implementing a mail example which shows how to use the Connectivity
 * service APIs to send e-mail.
 * The example provides a simple UI to compose an e-mail message and send it.
 * The post method uses
 * the Connectivity
 * service and the javax.mail API to send
 * the e-mail.
 */
public class MailServlet extends HttpServlet {
    @Resource(name = "mail/Session")
    private Session mailSession;
    private static final long serialVersionUID = 1L;
    private static final Logger LOGGER =
        LoggerFactory.getLogger(MailServlet.class);
    /** {@inheritDoc} */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        // Show input form to user
        response.setHeader("Content-Type", "text/html");
        PrintWriter writer = response.getWriter();
        writer.write("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
        Transitional//EN\" \"
        + \"http://www.w3.org/TR/html4/loose.dtd\">");
        writer.write("<html><head><title>Mail Test</title></head><body>");
        writer.write("<form action='\" method='post'>");
        writer.write("<table style='width: 100%'>");
        writer.write("<tr>");
        writer.write("<td width='100px'><label>From:</label></td>");
        writer.write("<td><input type='text' size='50' value='\"
        name='fromaddress'></td>");
```

```

        writer.write("</tr>");
        writer.write("<tr>");
        writer.write("<td><label>To:</label></td>");
        writer.write("<td><input type='text' size='50' value=''"
name='toaddress'></td>");
        writer.write("</tr>");
        writer.write("<tr>");
        writer.write("<td><label>Subject:</label></td>");
        writer.write("<td><textarea rows='1' cols='100' "
name='subjecttext'>Subject</textarea></td>");
        writer.write("</tr>");
        writer.write("<tr>");
        writer.write("<td><label>Mail:</label></td>");
        writer.write("<td><textarea rows='7' cols='100' name='mailtext'>Mail
Text</textarea></td>");
        writer.write("</tr>");
        writer.write("<tr>");
        writer.write("<tr>");
        writer.write("<tr>");
        writer.write("<td><input type='submit' value='Send Mail'></td>");
        writer.write("</tr>");
        writer.write("</table>");
        writer.write("</form>");
        writer.write("</body></html>");
    }
    /** {@inheritDoc} */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException,
        IOException {
        Transport transport = null;
        try {
            // Parse form parameters
            String from = request.getParameter("fromaddress");
            String to = request.getParameter("toaddress");
            String subjectText = request.getParameter("subjecttext");
            String mailText = request.getParameter("mailtext");
            if (from.isEmpty() || to.isEmpty()) {
                throw new RuntimeException("Form parameters From and To may
not be empty!");
            }
            // Construct message from parameters
            MimeMessage mimeMessage = new MimeMessage(mailSession);
            InternetAddress[] fromAddress = InternetAddress.parse(from);
            InternetAddress[] toAddresses = InternetAddress.parse(to);
            mimeMessage.setFrom(fromAddress[0]);
            mimeMessage.setRecipients(RecipientType.TO, toAddresses);
            mimeMessage.setSubject(subjectText, "UTF-8");
            MimeMultipart multiPart = new MimeMultipart("alternative");
            MimeBodyPart part = new MimeBodyPart();
            part.setText(mailText, "utf-8", "plain");
            multiPart.addBodyPart(part);
            mimeMessage.setContent(multiPart);
            // Send mail
            transport = mailSession.getTransport();
            transport.connect();
            transport.sendMessage(mimeMessage,
mimeMessage.getAllRecipients());
            // Confirm mail sending
            response.getWriter().println(
                "E-mail was sent (in local scenario stored in '<local-
server>/work/mailservice'"
                + " - in cloud scenario using configured mail
session).");
        } catch (Exception e) {
            LOGGER.error("Mail operation failed", e);
            throw new ServletException(e);
        } finally {
            // Close transport layer

```

```

        if (transport != null) {
            try {
                transport.close();
            } catch (MessagingException e) {
                throw new ServletException(e);
            }
        }
    }
}

```

4. Save the class.

3. Test the Application Locally

Test your code using the local file system before configuring your mail destination and testing the application in the cloud.

1. To test your application on the local server, select the servlet and choose **Run** > **Run As** > **Run on Server**.
2. Make sure that the *Manually define a new server* radio button is selected and select **SAP** > **Java Web Server**.
3. Choose *Finish*. A sender screen appears, allowing you to compose and send an e-mail. The sent e-mail is stored in the `work/mailservice` directory contained in the root of your SAP BTP local runtime server.

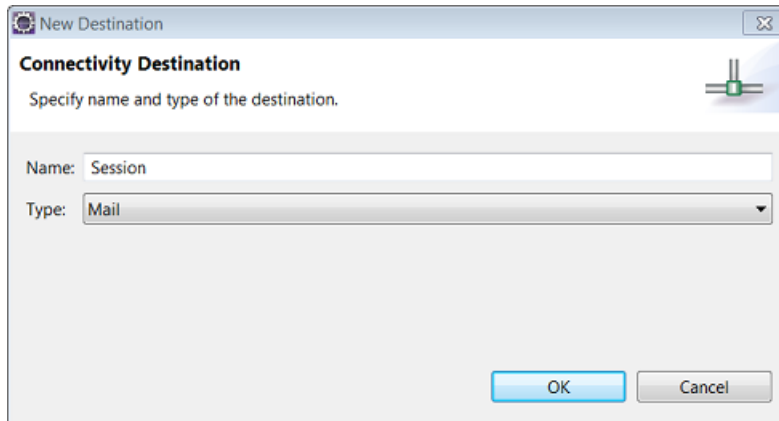
Note

To send the e-mail through a real e-mail server, you can configure a destination as described in the next section, but using the local server runtime. Remember that once you have configured a destination for local testing, messages are no longer sent to the local file system.

4. Test the Application in the Cloud

Create a mail destination that contains the SMTP settings of your e-mail provider. The name of the mail destination must match the name used in the resource reference in the `web.xml` descriptor.

1. In the Eclipse main menu, choose **File** > **New** > **Other** > **Server** > **Server**.
2. Select the server type *SAP Cloud Platform* and choose *Next*.
3. In the *SAP Cloud Platform Application* dialog box, enter the name of your application, subaccount, user, and password and choose *Finish*. The new server is listed in the *Servers* view.
4. Double-click the server and switch to the *Connectivity* tab.
5. In the *All Destinations* section, choose the **New Destination** button.
6. In the *New Destination* dialog box, enter the name **Session** and type **Mail** and choose *OK*.



New Destination

Connectivity Destination
Specify name and type of the destination.

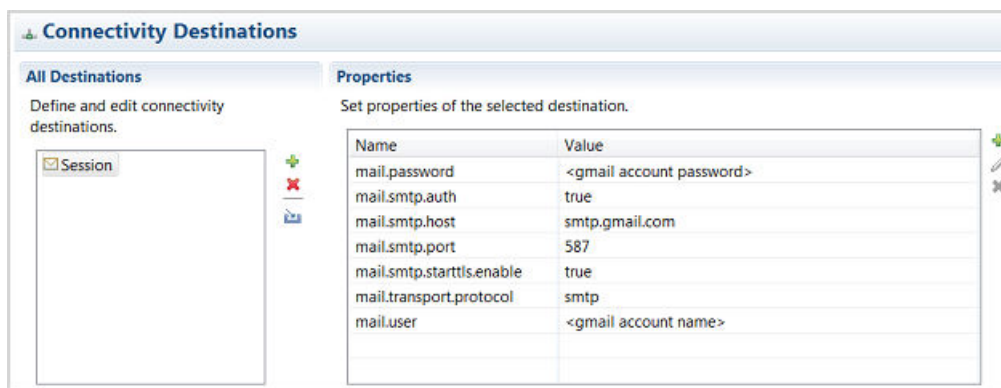
Name:

Type:

7. Configure the destination by adding the properties for port 587 (SMTP+STARTTLS) or 465 (SMTPS). To do this, choose the [Add Property](#) button in the [Properties](#) section:
 - To use port 587 (SMTP+STARTTLS), add the following properties:

Property	Value
mail.transport.protocol	smtp
mail.smtp.host	smtp.gmail.com
mail.smtp.auth	true
mail.smtp.starttls.enable	true
mail.smtp.port	587
mail.user	<gmail account name>
mail.password	<gmail account password>

The configured destination for port 587 is shown below:



Connectivity Destinations

All Destinations
Define and edit connectivity destinations.

☒ Session

Properties
Set properties of the selected destination.

Name	Value
mail.password	<gmail account password>
mail.smtp.auth	true
mail.smtp.host	smtp.gmail.com
mail.smtp.port	587
mail.smtp.starttls.enable	true
mail.transport.protocol	smtp
mail.user	<gmail account name>

- For port 465 (SMTPS), use the following properties:

Property	Value
mail.transport.protocol	smtps
mail.smtps.host	smtp.gmail.com

Property	Value
mail.smtps.auth	true
mail.smtps.port	465
mail.user	<gmail account name>
mail.password	<gmail account password>

8. Save the destination to upload it to the cloud. The settings take effect when the application is next started.
9. In the *Project Explorer* view, select `MailServlet.java` and choose **Run** **Run As** **Run on Server**.
10. Make sure that the *Choose an existing server* radio button is selected and select the server you have just defined.
11. Choose *Finish* to deploy to the cloud. You should now see the sender screen, where you can compose and send an e-mail

1.4.2 Consuming the Connectivity Service (HANA XS)

Create connectivity destinations for HANA XS applications, configure their security, add roles and test them in an enterprise or trial landscape.

- [Connectivity for SAP HANA XS \(Enterprise Version\) \[page 217\]](#)
- [Connectivity for SAP HANA XS \(Trial Version\) \[page 228\]](#)

Related Information

[Multitarget Applications for the Neo Environment Security](#)

1.4.2.1 Connectivity for SAP HANA XS (Enterprise Version)

Overview

This section represents the usage of the Connectivity service in a productive SAP HANA instance. Below are listed the scenarios depending on the connectivity and authentication types you use for your development work.

Connectivity Types

Internet Connectivity

In this case, you can develop an XS application in a productive SAP HANA instance at SAP BTP. This enables the application to connect to external Internet services or resources.

The corresponding XS parameters for all enterprise region hosts are the same (see also [Regions and Hosts Available for the Neo Environment](#)):

XS parameter	Value
useProxy	true
proxyHost	proxy
proxyPort	8080
useSSL	true / false

Note

In the outbound scenario, the `useSSL` property can be set to **true** or **false** depending on the XS application's needs.

For more information, see [Use XS Destinations for Internet Connectivity \[page 220\]](#)

On-Demand to On-Premise Connectivity

In this case, you can develop an XS application in a productive SAP HANA instance at SAP BTP. That way the application connects, via a Cloud Connector tunnel, to on-premise services and resources.

The corresponding XS parameters for all enterprise regions hosts are the same (see also [Regions and Hosts Available for the Neo Environment](#)):

XS parameter	Value
useProxy	true
proxyHost	localhost
proxyPort	20003
useSSL	false

Note

When XS applications consume the Connectivity service to connect to on-premise systems, the `useSSL` property must always be set to **false**.

The communication between the XS application and the proxy listening on localhost is always via HTTP. Whether the connection to the on-premise back end should be HTTP or HTTPS is a matter of access

control configuration in the Cloud Connector. For more information, see [Configure Access Control \(HTTP\) \[page 342\]](#).

For more information, see [Use XS Destinations for On-Demand to On-Premise Connectivity \[page 223\]](#)

Authentication Types

No Authentication

- Internet via HTTP - you can directly connect to an Internet service.
- Internet via HTTPS - you need to use SSL certificate to access an Internet service. To meet this requirement, proceed as follows:
 1. As a prerequisite, you need to have previously exported a certificate for the relevant HTTPS site.
 2. Then, open a Web browser and start the SAP HANA XS Administration Tool (<https://<schema><account>.<host>sap/hana/xs/admin/>).
 3. On the *XS Applications* page, expand the nodes in the application tree to locate your application.
 4. Select the `.xshttpdest` file to display details of the HTTP destination and then choose *Edit*.
 5. Select the *SSL Enabled* checkbox and in the *<Trust Store>* field choose your certificate from the drop-down list.
 6. As *SSL Authentication Type*, choose *Client Certificate*.
 7. In the *Authentication Type* section, leave the *None* radio button selected.
 8. Save your entries.

Basic Authentication

You need credentials to access an Internet or on-premise service. To meet this requirement, proceed as follows:

1. Open a Web browser and start the SAP HANA XS Administration Tool (<https://<schema><account>.<host>/sap/hana/xs/admin/>).
2. On the *XS Applications* page, expand the nodes in the application tree to locate your application.
3. Select the `.xshttpdest` file to display details of the HTTP destination and then choose *Edit*.
4. In the *AUTHENTICATION* section, choose the *Basic* radio button.
5. Enter the credentials for the on-premise service.
6. Save your entries.

1.4.2.1.1 Use XS Destinations for Internet Connectivity

Context

This section explains how to create a simple SAP HANA XS application, which is written in server-side JavaScript and makes use of theConnectivity service for making Internet connections.

In the HTTP example, the package is named **connectivity** and the XS application is **mapinfo**. The output displays information from Google Maps showing the distance between Frankfurt and Cologne, together with the consumed time if traveling with a car, as all this information is provided in American English..

📘 Note

You can check another outbound connectivity example (financial services that display the latest stock values) in *Developer Guide for SAP HANA Studio* → section "**8.4.1 Using the XSJS Outbound API**". For more information, see the SAP HANA Developer Guides listed in the *Related Links* section below. Refer to the [SAP BTP Release Notes](#) to find out which HANA SPS is supported by SAP BTP.

Prerequisites

- You have a productive SAP HANA instance. For more information, see [Using an SAP HANA XS Database System](#).
- You have installed the SAP HANA tools. For more information, see [Install SAP HANA Tools for Eclipse](#).

1. Initial Steps

To create and assign an XS destination, you need to have a developed HANA XS application.

- If you have already created one and have opened a database tunnel, go straight to procedure **2. Create an XS Destination File** on this page.
- If you need to create an XS application from scratch, go to page [Creating an SAP HANA XS Classic Hello World Application Using SAP HANA Studio](#) and execute procedures **1** to **4**. Then execute the procedures from this page (**2** to **5**).

📘 Note

The subpackage in which you will later create your XS destination and XSJS files has to be named **connectivity**.

2. Create an XS Destination File

1. In the *Project Explorer* view, select the *connectivity* folder and choose **File > New > File**.
2. Enter the file name **google.xshttpdest** and choose *Finish*.
3. Copy and paste the following destination configuration settings:

```
host = "maps.googleapis.com";
port = 80;
pathPrefix = "/maps/api/distancematrix/json";
authType = none;
useSSL = false;
timeout = 30000;
```

4. Save your changes.
5. Activate the file.

3. Create an XSJS File

1. In the *Project Explorer* view, select the *connectivity* folder and choose **File > New > File**.
2. Enter the file name **google_test.xsjs** and choose *Finish*.
3. Copy and paste the following JavaScript code into the file:

```
var destination_package = "connectivity.mapinfo";
var destination_name = "google";
try {
    var dest = $.net.http.readDestination(destination_package,
    destination_name);
    var client = new $.net.http.Client();
    var req = new $.web.WebRequest($.net.http.GET,
    "?origins=Frankfurt&destinations=Cologne&mode=driving&language=en-
    US&sensor=false");
    client.request(req, dest);
    var response = client.getResponse();

    $.response.contentType = "application/json";
    $.response.setBody(response.body.asString());
    $.response.status = $.net.http.OK;
} catch (e) {
    $.response.contentType = "text/plain";
    $.response.setBody(e.message);
}
```

4. Save your changes.
5. Activate the file.

Note

To consume an Internet service via HTTPS, you need to export your HTTPS service certificate into x.509 format, to import it into a trust store and to assign it to your activated destination. You need to do this in the SAP HANA XS Administration Tool (<https://<schema><account>.<host>/sap/hana/xs/admin/>). For more information, see *Developer Guide for SAP HANA Studio* → section "3.6.2.1 SAP HANA XS Application Authentication". For more information, see the SAP HANA Developer Guides listed in the *Related Links* section below. Refer to the [SAP BTP Release Notes](#) to find out which HANA SPS is supported by SAP BTP.

4. Grant a Role to the User

1. In the [Systems](#) view, expand [Security](#) > [Users](#) and then double-click your user ID.
2. On the [Granted Roles](#) tab, choose the [+ \(Add\)](#) button.
3. Select the **model_access** role in the list and choose [OK](#). The role is now listed on the [Granted Roles](#) tab.
4. Choose [Deploy](#) in the upper right corner of screen. A message confirms that your user has been modified.

5. Test Your Application

Open the cockpit and proceed as described in [Launch SAP HANA XS Applications](#).

You will be authenticated by SAML and should then see the following response:

```
{
  "destination_addresses" : [ "Cologne, Germany" ],
  "origin_addresses" : [ "Frankfurt, Germany" ],
  "rows" : [
    {
      "elements" : [
        {
          "distance" : {
            "text" : "190 km",
            "value" : 190173
          },
          "duration" : {
            "text" : "1 hour 58 mins",
            "value" : 7103
          },
          "status" : "OK"
        }
      ]
    }
  ],
  "status" : "OK"
}
```

Additional Example

You can also see an example for enabling server-side JavaScript applications to use the outbound connectivity API. For more information, see *Developer Guide for SAP HANA Studio* → section "8.4.1 Using the XSJS Outbound API".

📌 Note

For more information, see the SAP HANA Developer Guides listed below. Refer to the [SAP BTP Release Notes](#) to find out which HANA SPS is supported by SAP BTP.

See Also

- [Introduction to SAP HANA Development](#)
- [SAP HANA Web-Based Development Workbench](#)

Related Information

[XS Destination Properties \[page 226\]](#)

[Consume Internet Services \(Java Web or Java EE 6 Web Profile\) \[page 156\]](#)

[Creating an SAP HANA XS Hello World Application Using SAP HANA Web-based Development Workbench](#)

1.4.2.1.2 Use XS Destinations for On-Demand to On-Premise Connectivity

Context

This section explains how to create a simple SAP HANA XS application that consumes a sample back-end system exposed via the Cloud Connector.

In this example, the XS application consumes an on-premise system with basic authentication on landscape hana.ondemand.com.

Prerequisites

- You have a productive SAP HANA instance. For more information, see [Using an SAP HANA XS Database System](#).
- You have installed the SAP HANA tools. For more information, see [Install SAP HANA Tools for Eclipse](#). You need them to open a Database Tunnel.
- You have Cloud Connector 2.x installed on an on-premise system. For more information, see [Installation \[page 238\]](#).
- A sample back-end system with basic authentication is available on an on-premise host. For more information, see [Set Up an Application as a Sample Backend System \[page 190\]](#).
- You have created a tunnel between your subaccount and a Cloud Connector. For more information, see [Initial Configuration \[page 278\]](#) → section "Establishing Connections to SAP BTP".
- The back-end system is exposed for the SAP HANA XS application via Cloud Connector configuration using as settings: `virtual_host = virtualpingbackend` and `virtual_port = 1234`. For more information, see [Consume Backend Systems \(Java Web or Java EE 6 Web Profile\) \[page 171\]](#).

Note

The last two prerequisites can be achieved by exposing any other available HTTP service in your on-premise network. In this case, you shall adjust accordingly the `pathPrefix` value, mentioned below in procedure "2. Create an XS Destination File".

1. Initial Steps





To create and assign an XS destination, you need to have a developed HANA XS application.

- If you have already created one and have opened a database tunnel, go straight to procedure **2. Create an XS Destination File** on this page.
- If you need to create an XS application from scratch, go to page [Creating an SAP HANA XS Classic Hello World Application Using SAP HANA Studio](#) and execute procedures **1** to **4**. Then execute the procedures from this page (**2** to **5**).

Note

The subpackage in which you will later create your XS destination and XSJS files has to be named **connectivity**.

2. Create an XS Destination File

1. In the *Project Explorer* view, select the *connectivity* folder and choose  *File*  *New*  *File* .
2. Enter the file name **odop.xshttpdest** and choose *Finish*.
3. Copy and paste the following destination configuration settings:

```
host = "virtualpingbackend";
port = 1234;
useSSL = false;
pathPrefix = "/BackendAppHttpBasicAuth/basic";
useProxy = true;
proxyHost = "localhost";
proxyPort = 20003;
timeout = 3000;
```

Note

In case you use SDK with a version equal to or lower than **1.44.0.1** (Java Web) and **2.24.13** (Java EE 6 Web Profile) respectively, you should find the on-premise WAR files in directory `<SDK_location>/tools/samples/connectivity/onpremise`. Also, the `pathPrefix` should be `/PingAppHttpBasicAuth/pingbasic`.

4. Save your changes.
5. Activate the file.

3. Create an XSJS File

1. In the *Project Explorer* view, select the *connectivity* folder and choose **File > New > File**.
2. Enter the file name **ODOPTest.xsjs** and choose *Finish*.
3. Copy and paste the following JavaScript code into the file:

```
$.response.contentType = "text/html";
var dest = $.net.http.readDestination("connectivity", "odop");
var client = new $.net.http.Client();
var req = new $.web.WebRequest($.net.http.GET, "");
client.request(req, dest);
var response = client.getResponse().body.asString();
$.response.setBody(response);
```

4. Save your changes.
5. Activate the file.

Note

You also need to enter your on-premise credentials. You should not enter them in the destination file since they must not be exposed as plain text.

4. Provide Secured Credentials

1. Open a Web browser and start the SAP HANA XS Administration Tool (<https://<schema><account>.<host>/sap/hana/xs/admin/>).
2. On the *XS Applications* page, expand the nodes in the application tree to locate your application.
3. Select the *odop.xshttpdest* file to display the HTTP destination details and then choose *Edit*.
4. In section *AUTHENTICATION*, choose the *Basic* radio button.
5. Enter your on-premise credentials (user and password).
6. Save your entries.

Note

If you later need to make another configuration change to your XS destination, you need to enter your password again since it is no longer remembered by the editor.

5. Grant a Role to the User

1. In the *Systems* view, expand **Security > Users** and then double-click your user ID.
2. On the *Granted Roles* tab, choose the **+ (Add)** button.
3. Select the **model_access** role in the list and choose *OK*. The role is now listed on the *Granted Roles* tab.
4. Choose *Deploy* in the upper right corner of screen. A message confirms that your user has been modified.

6. Test Your Application

Open the cockpit and proceed as described in [Launch SAP HANA XS Applications](#).

Principal Propagation to on-premise systems

Principal Propagation scenario is available for HANA XS applications. It is used for propagating the currently logged in user to an on-premise back-end system using the Cloud Connector and connectivity service. To configure the scenario make sure to:

1. Create an XS destination with the parameter `authType=SamlAssertionPropagation`.
2. Open the Cloud Connector and mark your HANA instance as trusted in the *Principal Propagation* tab. The HANA instance name is displayed in the cockpit under ► *SAP HANA/SAP ASE* ► *Databases & Schemas* ►. For more information, see [Set Up Trust \[page 304\]](#).

Related Information

[XS Destination Properties \[page 226\]](#)

[Consume Backend Systems \(Java Web or Java EE 6 Web Profile\) \[page 171\]](#)

[Creating an SAP HANA XS Hello World Application Using SAP HANA Web-based Development Workbench](#)

1.4.2.1.3 XS Destination Properties

XS Property	Description	Value
host	It enables you to specify the host name of the HTTP destination providing the service or data you want your SAP HANA XS application to access.	URL (string)
port	It enables you to specify the port number to use for connections to the HTTP destination hosting the service or data you want your SAP HANA XS application to access.	<ul style="list-style-type: none">• For Internet connection: 80, 443• For on-demand to on-premise connection: 1080• For service-to-service connection: 8443
pathPrefix	It enables you to specify a text element to add to the start of the URL used for connections to the service specified in the HTTP destination configuration.	"..."; (string)

XS Property	Description	Value
useProxy	<p>It enables you to specify whether a proxy server must be used to resolve the host name specified in the HTTP destination configuration file.</p> <p>Depends on the authentication type and the landscape on which you deploy your SAP HANA XS application.</p>	<ul style="list-style-type: none"> For all productive landscapes: <ul style="list-style-type: none"> Internet: true On-premise: true Service-to-service: false <div> Note See also: Connectivity for SAP HANA XS (Enterprise Version) [page 217] </div>
proxyHost	<p>Depending on the authentication type and the landscape on which you deploy your SAP HANA XS application.</p>	<ul style="list-style-type: none"> For all productive landscapes: <ul style="list-style-type: none"> Internet: proxy On-premise: localhost <div> Note See also: Connectivity for SAP HANA XS (Enterprise Version) [page 217] </div>
proxyPort	<p>Depending on the authentication type and the landscape on which you deploy your SAP HANA XS application.</p>	<ul style="list-style-type: none"> For all productive landscapes: <ul style="list-style-type: none"> Internet: 8080 On-premise: 20003 Service-to-service: 8080 <div> Note See also: Connectivity for SAP HANA XS (Enterprise Version) [page 217] </div>
authType	<p>It enables you to specify the authentication method that must be used for connection requests for the service located at the HTTP destination specified in the configuration.</p>	none, basic, AssertionTicket, SAML Assertion Propagation
useSSL	<p>It is of type Boolean and enables you to specify whether the outbound connection between SAP HANA XS and the HTTP destination is secured with the Secure Sockets Layer (SSL) protocol (HTTPS).</p>	true, false <div> Note See also: Connectivity for SAP HANA XS (Enterprise Version) [page 217] </div>
timeout	<p>It enables you to specify for how long (in milliseconds) an application tries to connect to the remote host specified in the HTTP destination configuration.</p>	-1 (default) In the connectivity examples: 3000

Related Information

- [HTTP Destination Configuration Syntax](#)
- [Regions](#)

1.4.2.2 Connectivity for SAP HANA XS (Trial Version)

Context

This section represents the usage of the Connectivity service when you develop and deploy SAP HANA XS applications in a trial environment. Currently, you can make XS destinations for consuming HTTP Internet services only.

The procedure below lets you create a simple SAP HANA XS application which is written in server-side JavaScript and makes use of the Connectivity service for making Internet connections. In the HTTP example, the package is named **connectivity** and the XS application is **mapinfo**. The output displays information from Google Maps showing the distance between Frankfurt and Cologne, together with the consumed time if traveling with a car, as all this information is provided in American English.

Features

In this case, you can develop an XS application in a trial environment at SAP BTP so that the application connects to external Internet services or resources.

XS parameter	hanatrial.ondemand.com
useProxy	true
proxyHost	proxy-trial
proxyPort	8080
useSSL	true / false

📘 Note

The `useSSL` property can be set to **true** or **false** depending on the XS application's needs.

1. Initial Steps

To create and assign an XS destination, you need to have a developed HANA XS application.

- If you have already created one and have opened a database tunnel, go straight to procedure **2. Create an XS Destination File** on this page.
- If you need to create an XS application from scratch, go to page [Creating an SAP HANA XS Classic Hello World Application Using SAP HANA Studio](#) and execute procedures **1** to **4**. Then execute the procedures from this page (**2** to **5**).

📌 Note

The subpackage in which you will later create your XS destination and XSJS files has to be named **connectivity**.

2. Create an XS Destination File

1. In the *Project Explorer* view, select the *connectivity* folder and choose **File > New > File**.
2. Enter the file name **google.xshttpdest** and choose *Finish*.
3. Copy and paste the following destination configuration settings:

```
host = "maps.googleapis.com";
port = 80;
pathPrefix = "/maps/api/distancematrix/json";
useProxy = true;
proxyHost = "proxy-trial";
proxyPort = 8080;
authType = none;
useSSL = false;
timeout = 30000;
```

4. Save your changes.
5. Activate the file.

3. Create an XSJS File

1. In the *Project Explorer* view, select the *connectivity* folder and choose **File > New > File**.
2. Enter the file name **google_test.xsjs** and choose *Finish*.
3. Copy and paste the following JavaScript code into the file:

```
var destination_package = "connectivity.mapinfo";
var destination_name = "google";
try {
    var dest = $.net.http.readDestination(destination_package,
destination_name);
    var client = new $.net.http.Client();
    var req = new $.web.WebRequest($.net.http.GET,
"?origins=Frankfurt&destinations=Cologne&mode=driving&language=en-
US&sensor=false");
    client.request(req, dest);
    var response = client.getResponse();

$.response.contentType = "application/json";
$.response.setBody(response.body.asString());
$.response.status = $.net.http.OK;
```

```

    } catch (e) {
        $.response.contentType = "text/plain";
        $.response.setBody(e.message);
    }
}

```

4. Save your changes.
5. Activate the file.

4. Grant a Role to the User

1. In the [Systems](#) view, select your system and from the context menu choose [SQL Console](#).
2. In the SQL console, enter the following, replacing <SAP HANA Cloud user> with your user:

```

call
'HCP'.HCP_GRANT_ROLE_TO_USER('p1234567890trial.myhanaxs.hello::model_access'
, '<SAP HANA Cloud user>')

```

3. Execute the procedure. You should see a confirmation that the statement was successfully executed.

5. Test Your Application

Open the cockpit and proceed as described in [Launch SAP HANA XS Applications](#).

You will be authenticated by SAML and should then see the following response:

```

{
  "destination_addresses" : [ "Cologne, Germany" ],
  "origin_addresses" : [ "Frankfurt, Germany" ],
  "rows" : [
    {
      "elements" : [
        {
          "distance" : {
            "text" : "190 km",
            "value" : 190173
          },
          "duration" : {
            "text" : "1 hour 58 mins",
            "value" : 7103
          },
          "status" : "OK"
        }
      ]
    }
  ],
  "status" : "OK"
}

```

Related Information

[Creating an SAP HANA XS Hello World Application Using SAP HANA Web-based Development Workbench](#)

1.4.3 Consuming the Destination Configuration Service

Retrieve destination configurations for your cloud application in the Neo environment, in a secure and reliable way.

Using the Destination Configuration service, you can create, edit, update and read destinations, keystores and certificates on application, subaccount, or subscription level, see [Managing Destinations \[page 54\]](#). You can access these destinations through your application at runtime or from the SAP BTP cockpit, see [Configure Destinations from the Cockpit \[page 76\]](#).

Prerequisites

You must have administrative access to a subaccount within the Neo environment.

Required Credentials

The Destination Configuration service requires OAuth 2.0 credentials for all REST API methods. To manage and read destinations and certificates, you must create an OAuth client and assign any of the following permissions: *Manage Destinations* (read/write) or *Read Destination* (read only), see [Using Platform APIs](#).

Destination Configuration Service REST API

The Destination Configuration service provides a REST API, which lets you configure the destinations that you need to connect your application to another system or service, see [SAP Business Accelerator Hub](#).

For the `create` and `update` methods, you must send the destination values as a `properties` file with the `multipart/form-data` media type as form data.

Note

When you read a destination, for security reasons the same file is downloaded without the sensitive properties.

Example: Create and Read Destination Configuration Requests on Tenant Level

Sample Code

```
curl -X POST https://api.{landscape_domain}.configuration/api/rest/
oauth/SPACES/{account}/connectivity/ -H 'authorization: Bearer
{access_token}' -H 'content-type: multipart/form-data' -F
'{configuration_name}=@{configuration_file_path}' -v
curl -O GET https://api.{landscape_domain}.configuration/api/rest/oauth/
SPACES/{account}/connectivity/{configuration_name} -H 'authorization: Bearer
{access_token}' -H 'content-type: application/octet-stream'
```

Destination Configuration Properties

Depending on the authentication type used, different properties are required for a destination. Find the available properties for each authentication type in [HTTP Destinations \[page 91\]](#).

1.5 Cloud Connector (Neo Environment)

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

Note

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

- [Context \[page 233\]](#)
- [Basic Scenarios \[page 234\]](#)
- [Basic Tasks \[page 237\]](#)
- [Advantages \[page 234\]](#)
- [Extended Scenarios \[page 236\]](#)
- [What's New? \[page 237\]](#)

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	

- [Configuration \[page 277\]](#)
- [Installation \[page 238\]](#)
- [Administration \[page 516\]](#)
- [Frequently Asked Questions \[page 604\]](#)
- [Upgrade \[page 600\]](#)
- [Security \[page 589\]](#)
- [Update the Java VM \[page 602\]](#)
- [Uninstallation \[page 603\]](#)
- [Connectivity Support \[page 615\]](#)
- [Security Guidelines \[page 596\]](#)

Context

The Cloud Connector:

- Serves as a link between SAP BTP applications and on-premise systems.
 - Combines an easy setup with a clear configuration of the systems that are exposed to the SAP BTP.
 - Lets you use existing on-premise assets without exposing the entire internal landscape.
- Runs as on-premise agent in a secured network.
 - Acts as a reverse invoke proxy between the on-premise network and SAP BTP.
- Provides fine-grained control over:
 - On-premise systems and resources that can be accessed by cloud applications.
 - Cloud applications using the Cloud Connector.
- Lets you use the features that are required for business-critical enterprise scenarios.
 - Recovers broken connections automatically.
 - Provides audit logging of inbound traffic and configuration changes.

- Can be run in a high-availability setup.

Caution

The Cloud Connector must not be used to connect to products other than SAP BTP or S/4HANA Cloud.

Back to [Content \[page 232\]](#)

Advantages

Compared to the approach of opening ports in the firewall and using reverse proxies in the DMZ to establish access to on-premise systems, the Cloud Connector offers the following benefits:

- You don't need to configure the on-premise firewall to allow external access from SAP BTP to internal systems. For allowed outbound connections, no modifications are required.
- The Cloud Connector supports HTTP as well as additional protocols. For example, the RFC protocol supports native access to ABAP systems by invoking function modules.
- You can use the Cloud Connector to connect on-premise databases or BI tools to SAP HANA databases in the cloud.
- The Cloud Connector lets you propagate the identity of cloud users to on-premise systems in a secure way.
- Easy installation and configuration, which means that the Cloud Connector comes with a low TCO and is tailored to fit your cloud scenarios.
- SAP provides standard support for the Cloud Connector.

Back to [Content \[page 232\]](#)

Basic Scenarios

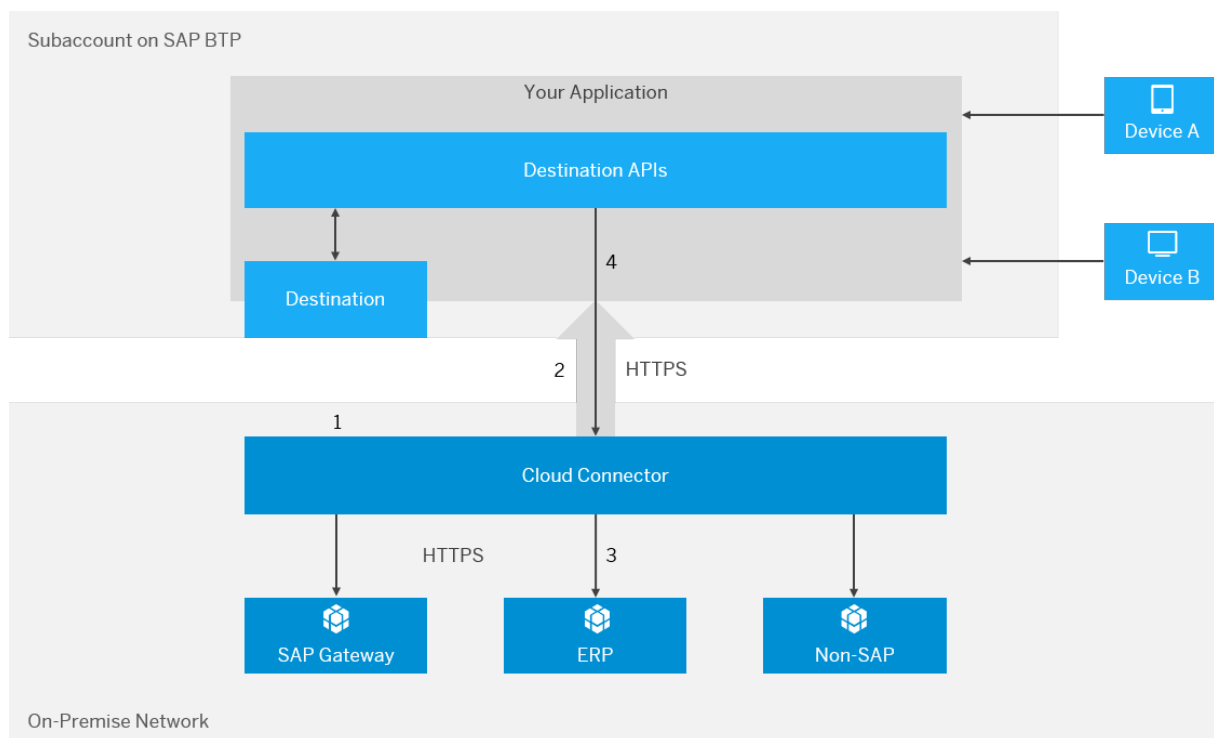
[Connecting Cloud Applications to On-Premise Systems \[page 234\]](#)

[Connecting On-Premise Database Tools to SAP HANA Databases \[page 235\]](#)

Note

This section refers to the Cloud Connector installation in a standard on-premise network. Find setup options for other system environments in [Extended Scenarios \[page 236\]](#).

Connecting Cloud Applications to On-Premise Systems

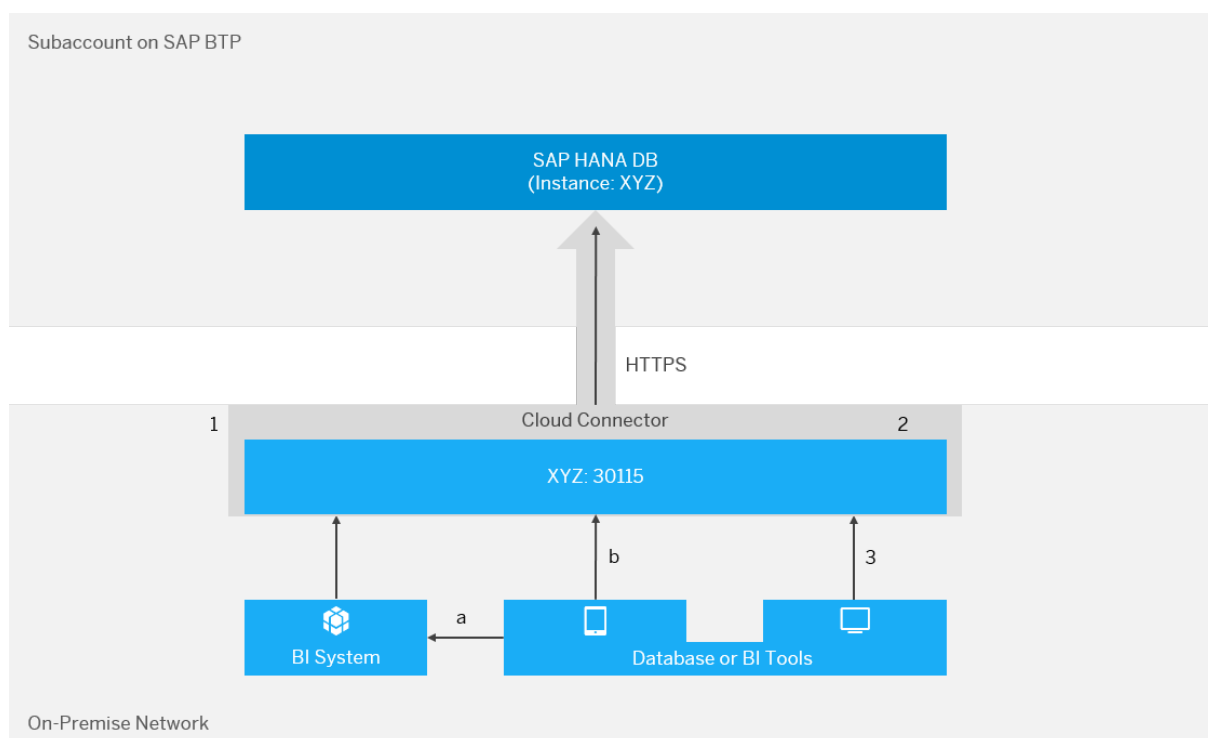


1. Install the Cloud Connector: [Installation \[page 238\]](#)
2. Set up the connection between Cloud Connector, back-end system and your SAP BTP subaccount : [Initial Configuration \[page 278\]](#), [Managing Subaccounts \[page 291\]](#)
3. Allow your cloud application to access a back-end system on the intranet: [Configure Access Control \[page 340\]](#)
4. Connect your cloud application to an on-premise system:
[Consuming the Connectivity Service \[page 135\]](#) (Neo environment)

Back to [Basic Scenarios \[page 234\]](#)

Back to [Content \[page 232\]](#)

Connecting On-Premise Database Tools to SAP HANA Databases



1. Install and configure the Cloud Connector: [Installation \[page 238\]](#), [Initial Configuration \[page 278\]](#), [Managing Subaccounts \[page 291\]](#)
2. Access HANA databases on SAP BTP: [Configure a Service Channel for an SAP HANA Database \[page 492\]](#)
3. Connect on-premise database or BI tools to a HANA database on SAP BTP: [Connect DB Tools to SAP HANA via Service Channels \[page 495\]](#)

Note

You can use service channels also for other purposes:

- Connect to a virtual machine on SAP BTP.
- Configure an RFC connection from your on-premise system to S/4HANA Cloud.

See [Using Service Channels \[page 491\]](#).


Back to [Basic Scenarios \[page 234\]](#)

Back to [Content \[page 232\]](#)

Extended Scenarios

Besides the standard setup: *SAP BTP - Cloud Connector - on-premise system/network*, you can also use the Cloud Connector to connect SAP BTP applications to other cloud-based environments, as long as they are operated in a way that is comparable to an on-premise network from a functional perspective. This is particularly true for infrastructure (IaaS) hosting solutions.

Here's an overview of all environments in which you can or cannot set up the Cloud Connector:

Cloud Connector	Environment	Examples
Can be set up in: <div> Note Within extended scenarios that allow a Cloud Connector setup, special procedures may apply for configuration. If so, they are mentioned in the corresponding configuration steps.</div>	Customer on-premise network (see Basic Scenarios [page 234])	SAP ERP, SAP S/4HANA
	SAP Hosting	SAP HANA Enterprise Cloud (HEC)
	Third-party IaaS providers (hosting)	Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP)
Cannot be set up in:	SAP SaaS solutions	SAP SuccessFactors, SAP Concur, SAP Ariba
	SAP cloud-based enterprise solutions	SAP S/4HANA Cloud, SAP C/4HANA
	Third-party PaaS providers	AWS, Azure, GCP
	Third-party SaaS providers	

Back to [Content \[page 232\]](#)

Basic Tasks

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

- Install the Cloud Connector: [Installation \[page 238\]](#)
- Perform the initial configuration for the Cloud Connector: [Initial Configuration \[page 278\]](#)
- Register the Cloud Connector for your SAP BTP subaccount: [Managing Subaccounts \[page 291\]](#)

Back to [Content \[page 232\]](#)

What's New?

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

Back to [Content \[page 232\]](#)

Related Information

[Installation \[page 238\]](#)
[Configuration \[page 277\]](#)
[Administration \[page 516\]](#)
[Security \[page 589\]](#)
[Upgrade \[page 600\]](#)
[Update the Java VM \[page 602\]](#)
[Uninstallation \[page 603\]](#)
[Frequently Asked Questions \[page 604\]](#)

1.5.1 Installation

Choose a procedure to install the Cloud Connector on your operating system.

Portable Version vs. Installer Version

On **Microsoft Windows** and **Linux**, two installation modes are available: a `portable` version and an `installer` version. On **Mac OS X**, only the `portable` version is available.

- `portable` version: can be installed easily, by extracting a compressed archive into an empty directory. It does not require administrator or root privileges for the installation, and you can run multiple instances on the same host.
Restrictions:
 - You cannot run it in the background as a Windows Service or Linux daemon (with automatic start capabilities at boot time).
 - The portable version does not support an automatic upgrade procedure. To update a portable installation, you must delete the current one, extract the new version, and then re-do the configuration.
 - Portable versions are meant for non-productive scenarios only.
 - The environment variable `JAVA_HOME` is relevant when starting the instance, and therefore must be set properly.
- `installer` version: requires administrator or root permissions for the installation and can be set up to run as a Windows service or Linux daemon in the background. You can upgrade it easily, retaining all the configuration and customizing.

Note

We strongly recommend that you use this variant for a productive setup.

Caution

Cloud Connector is based on Tomcat.

Tomcat is a well-known and well-documented server, whose configuration can be adapted to one's own needs. However, we strongly recommend that you **do not modify** its configuration files like

`conf\server.xml` with a text editor, because the Cloud Connector is making assumptions about the content of those files.

If you still want to do custom modifications, you do so at your own risk. In this case, we can no longer guarantee that the Cloud Connector keeps working as expected. For any issues that can be traced back to such changes, we cannot provide support, in particular, if such changes cause trouble during upgrade to a newer version.

Prerequisites

- There are some general prerequisites you must fulfill to successfully install the Cloud Connector, see [Prerequisites \[page 239\]](#).
- For OS-specific requirements and procedures, see section **Tasks** below.

Tasks

- [Installation on Microsoft Windows OS \[page 265\]](#)
- [Installation on Linux OS \[page 268\]](#)
- [Installation on Mac OS X \[page 271\]](#)

Related Information

[Sizing Recommendations \[page 258\]](#)

[Recommendations for Secure Setup \[page 272\]](#)

[Uninstallation \[page 603\]](#)

1.5.1.1 Prerequisites

Prerequisites for successful installation of the Cloud Connector.

Content

Section	Description
Connectivity Restrictions [page 240]	General information about SAP BTP and connectivity restrictions.
Hardware [page 240]	Hardware prerequisites for a physical or virtual machine.
Software [page 241]	Required software download and installation.
JDKs [page 241]	Java Development Kit (JDK) versions that you can use.
Product Availability Matrix [page 241]	Availability of operating systems/versions for specific Cloud Connector versions.
Network [page 242]	Required Internet connection to SAP BTP hosts per region.

Note

For additional system requirements, see also [System Requirements \[page 256\]](#).

Connectivity Restrictions

For specific information about all Connectivity restrictions, see [Connectivity for the Neo Environment: Restrictions \[page 5\]](#).

Back to [Content \[page 239\]](#)

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 \[page 239\]](#)

Software



- You have downloaded the Cloud Connector installation archive from [SAP Development Tools for Eclipse](#).
- A full JDK must be installed. Lightweight JRE installations are not sufficient. You can download a fitting up-to-date SAP JVM from [SAP Development Tools for Eclipse](#).

⚠ Caution

Do not use [Apache Portable Runtime \(APR\)](#)  on the system on which you use the Cloud Connector. If you cannot avoid this restriction and want to use APR at your own risk, you must manually adapt the `server.xml` configuration file in directory `<scc_installation_folder>/conf`. To do so, follow the steps in [HTTPS port configuration](#)  for APR.

[Back to Content \[page 239\]](#)

JDKs

JDK	Version	Cloud Connector Version
SAP JVM 64-bit (recommended)	7	2.x up to 2.12.2
	8	2.7.2 and higher
Oracle JDK 64-bit	7	2.x up to 2.12.2
	8	2.7.2 and higher
SAP Machine  64-bit	11	2.14.0 and higher
SAP Machine  64-bit	17	2.15.0 and higher

[Back to Content \[page 239\]](#)

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

Operating System Version	Architecture	Cloud Connector Version
SUSE Linux Enterprise Server 12, Red Hat Enterprise Linux 7	x86_64	2.5.1 and higher
Mac OS X 10.9 (Mavericks), Mac OS X 10.10 (Yosemite)	x86_64	2.5.1 and higher
Windows 10	x86_64	2.7.2 and higher
Mac OS X 10.11 (El Capitan)	x86_64	2.8.1 and higher
Windows Server 2016	x86_64	2.9.1 and higher
Windows Server 2019, Mac OS X 10.12 (Sierra), Mac OS X 10.13 (High Sierra), Mac OS X 10.14 (Mojave)	x86_64	2.11.3 and higher
SUSE Linux Enterprise Server 15	x86_64	2.12.0 and higher
Red Hat Enterprise Linux 8	x86_64	2.12.2 and higher
SUSE Linux Enterprise Server 12, SUSE Linux Enterprise Server 15, Red Hat Enterprise Linux 7, Red Hat Enterprise Linux 8	ppc64le	2.13.0 and higher
Windows Server 2022, macOS 10.15 (Catalina)	x86_64	2.14.0 and higher
Windows 11, Red Hat Enterprise Linux 9	x86_64	2.15.0 and higher
Red Hat Enterprise Linux 9	ppc64le	2.15.0 and higher
Oracle Linux 8	x86_64	2.15.2 and higher
Oracle Linux 9, macOS 11 (Big Sur), macOS 12 (Monterey), macOS 13 (Ventura)	x86_64	2.16.0 and higher
macOS 11 (Big Sur), macOS 12 (Monterey), macOS 13 (Ventura)	aarch64	2.16.0 and higher

Back to [Content \[page 239\]](#)

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.

Note

For general information on IP ranges per region, see [Regions](#) (Cloud Foundry and ABAP environment) or [Regions and Hosts Available for the Neo Environment](#). Find detailed information about the region status and planned network updates on [Platform Updates and Notifications](#).

[Cloud Foundry Environment \[page 243\]](#)

[ABAP Environment \[page 251\]](#)

[Neo Environment \[page 252\]](#)

[Trial \[page 254\]](#)

Region (Region Host)	Hosts	IP Address
Cloud Foundry Environment		
Note In the Cloud Foundry environment, IPs are controlled by the respective IaaS provider - Amazon Web Services (AWS), Microsoft Azure (Azure), or Google Cloud. IPs may change due to network updates on the provider side. Any planned changes will be announced at least 4 weeks before they take effect. See also Regions .		
Europe (Frankfurt) - AWS (cf.eu10.hana.ondemand.com) <i>Enterprise & Trial</i>	connectivitynotification.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223 Additional IP addresses (valid starting from late Q1 2024): 18.159.31.22, 3.69.186.98, 3.77.195.119
	connectivitycertsigning.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223 Additional IP addresses (valid starting from late Q1 2024): 18.159.31.22, 3.69.186.98, 3.77.195.119
	connectivitytunnel.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223 Additional IP addresses (valid starting from late Q1 2024): 18.159.31.22, 3.69.186.98, 3.77.195.119
Caution Additional IP addresses were added to this region. Action: If you restrict system access by <i>allowlisting</i> IPs in firewall rules, make sure you update your configuration accordingly. The new IP addresses may be used starting from end of Q1, 2024.		

Region (Region Host)	Hosts	IP Address
	connectivitytunnel.cf.eu10-002.hana.ondemand.com	3.64.227.236, 3.126.229.22, 18.193.180.19 Additional IP addresses (valid starting from late Q1 2024): 18.153.123.11, 3.121.37.195, 3.73.215.90
	connectivitytunnel.cf.eu10-003.hana.ondemand.com	3.127.77.3, 3.64.196.58, 18.156.151.247 Additional IP addresses (valid starting from late Q1 2024): 18.197.252.154, 3.79.137.29, 52.58.93.50
	connectivitytunnel.cf.eu10-004.hana.ondemand.com	3.65.185.47, 3.70.38.218, 18.196.206.8 Additional IP addresses (valid starting from late Q1 2024): 3.73.109.100, 3.73.8.210, 52.59.18.183
	connectivitynotification.cf.eu11.hana.ondemand.com	3.124.207.41, 18.157.105.117, 18.156.209.198 Additional IP addresses (valid starting from late Q1 2024): 3.66.26.249, 3.72.216.204, 3.74.99.245
Europe (Frankfurt) - AWS (cf.eu11.hana.ondemand.com)	connectivitycertsigning.cf.eu11.hana.ondemand.com	3.124.207.41, 18.157.105.117, 18.156.209.198 Additional IP addresses (valid starting from late Q1 2024): 3.66.26.249, 3.72.216.204, 3.74.99.245

⚠ Caution

Additional IP addresses were added to this region.

Action:

If you restrict system access by *allowlisting* IPs in firewall rules, make sure you update your configuration accordingly. The new IP addresses may be used starting from end of Q1, 2024.

Region (Region Host)	Hosts	IP Address
	connectivitytunnel.cf.eu11.hana.ondemand.com	3.124.207.41, 18.157.105.117, 18.156.209.198 Additional IP addresses (valid starting from late Q1 2024): 3.66.26.249, 3.72.216.204, 3.74.99.245
Europe (Netherlands) - Azure (cf.eu20.hana.ondemand.com)	connectivitynotification.cf.eu20.hana.ondemand.com	40.119.153.88
	connectivitycertsigning.cf.eu20.hana.ondemand.com	40.119.153.88
	connectivitytunnel.cf.eu20.hana.ondemand.com	40.119.153.88
	connectivitytunnel.cf.eu20-001.hana.ondemand.com	20.82.83.59
Europe (Frankfurt) - Google Cloud (cf.eu30.hana.ondemand.com)	connectivitynotification.cf.eu30.hana.ondemand.com	35.198.143.110
	connectivitycertsigning.cf.eu30.hana.ondemand.com	35.198.143.110
	connectivitytunnel.cf.eu30.hana.ondemand.com	35.198.143.110
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 Additional IP addresses (valid starting from late Q1 2024): 18.213.242.208, 3.214.110.153, 34.205.56.51
	connectivitycertsigning.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211 Additional IP addresses (valid starting from late Q1 2024): 18.213.242.208, 3.214.110.153, 34.205.56.51

Enterprise & Trial

⚠ Caution

Additional IP addresses were added to this region.

Action:

If you restrict system access by *allowlisting IPs* in firewall rules, make sure you update your configuration accordingly. The new IP addresses may be used starting from end of Q1, 2024.

Region (Region Host)	Hosts	IP Address
	connectivitytunnel.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211 Additional IP addresses (valid starting from late Q1 2024): 18.213.242.208, 3.214.110.153, 34.205.56.51
	connectivitytunnel.cf.us10-001.hana.ondemand.com	3.220.114.17, 3.227.182.44, 52.86.131.53 Additional IP addresses (valid starting from late Q1 2024): 44.218.82.203, 44.219.57.163, 50.16.106.103
	connectivitytunnel.cf.us10-002.hana.ondemand.com	34.202.68.0, 54.234.152.59, 107.20.66.86 Additional IP addresses (valid starting from late Q1 2024): 3.214.116.95, 54.144.230.36, 54.226.37.104
US West (WA) - Azure (cf.us20.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 East (VA) - Azure (cf.us21.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
US Central (IA) - Google Cloud (cf.us30.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

Region (Region Host)	Hosts	IP Address
Brazil (São Paulo) - AWS (cf.br10.hana.ondemand.com)	connectivitynotification.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4, 54.232.179.204 Additional IP addresses (valid starting from late Q1 2024): 18.228.53.198, 52.67.149.240, 54.94.179.209
	connectivitycertsigning.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4, 54.232.179.204 Additional IP addresses (valid starting from late Q1 2024): 18.228.53.198, 52.67.149.240, 54.94.179.209
	connectivitytunnel.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4, 54.232.179.204 Additional IP addresses (valid starting from late Q1 2024): 18.228.53.198, 52.67.149.240, 54.94.179.209
Japan (Tokyo) - AWS (cf.jp10.hana.ondemand.com)	connectivitynotification.cf.jp10.hana.ondemand.com	13.114.117.83, 3.114.248.68, 3.113.252.15 Additional IP addresses (valid starting from late Q1 2024): 18.178.155.134, 57.180.140.5, 57.180.145.179

Region (Region Host)	Hosts	IP Address
<p>uration accordingly. The new IP addresses may be used starting from end of Q1, 2024.</p>	connectivitycertsigning.cf.jp10.hana.ondemand.com	<p>13.114.117.83, 3.114.248.68, 3.113.252.15</p> <p>Additional IP addresses (valid starting from late Q1 2024):</p> <p>18.178.155.134, 57.180.140.5, 57.180.145.179</p>
	connectivitytunnel.cf.jp10.hana.ondemand.com	<p>13.114.117.83, 3.114.248.68, 3.113.252.15</p> <p>Additional IP addresses (valid starting from late Q1 2024):</p> <p>18.178.155.134, 57.180.140.5, 57.180.145.179</p>
<p>Japan (Tokyo) - Azure (cf. jp20.hana.ondemand.com)</p>	connectivitynotification.cf.jp20.hana.ondemand.com	20.43.89.91
	connectivitycertsigning.cf.jp20.hana.ondemand.com	20.43.89.91
	connectivitytunnel.cf.jp20.hana.ondemand.com	20.43.89.91
<p>Australia (Sydney) - AWS (cf. ap10.hana.ondemand.com)</p>	connectivitynotification.cf.ap10.hana.ondemand.com	<p>13.236.220.84, 13.211.73.244, 3.105.95.184</p> <p>Additional IP addresses (valid starting from late Q1 2024):</p> <p>13.55.188.95, 3.105.212.249, 3.106.45.106</p>
	connectivitycertsigning.cf.ap10.hana.ondemand.com	<p>13.236.220.84, 13.211.73.244, 3.105.95.184</p> <p>Additional IP addresses (valid starting from late Q1 2024):</p> <p>13.55.188.95, 3.105.212.249, 3.106.45.106</p>

⚠ Caution

Additional IP addresses were added to this region.

Action:

If you restrict system access by *allowlisting* IPs in firewall rules, make sure you update your configuration accordingly. The new IP addresses may be used starting from end of Q1, 2024.

Region (Region Host)	Hosts	IP Address
Asia Pacific (Singapore) - AWS (cf.ap11.hana.ondemand.com)	connectivitytunnel.cf.ap10.hana.ondemand.com	13.236.220.84, 13.211.73.244, 3.105.95.184 Additional IP addresses (valid starting from late Q1 2024): 13.55.188.95, 3.105.212.249, 3.106.45.106
	connectivitynotification.cf.ap11.hana.ondemand.com	3.0.9.102, 18.140.39.70, 18.139.147.53 Additional IP addresses (valid starting from late Q1 2024): 13.229.158.122, 18.140.228.217, 52.74.215.89
	connectivitycertsigning.cf.ap11.hana.ondemand.com	3.0.9.102, 18.140.39.70, 18.139.147.53 Additional IP addresses (valid starting from late Q1 2024): 13.229.158.122, 18.140.228.217, 52.74.215.89
	connectivitytunnel.cf.ap11.hana.ondemand.com	3.0.9.102, 18.140.39.70, 18.139.147.53 Additional IP addresses (valid starting from late Q1 2024): 13.229.158.122, 18.140.228.217, 52.74.215.89
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 Additional IP addresses (valid starting from late Q1 2024): 13.209.236.215, 43.201.194.105, 43.202.204.5

⚠ Caution

Additional IP addresses were added to this region.

Action:


If you restrict system access by *allowlisting* IPs in firewall rules, make sure you update your configuration accordingly. The new IP addresses may be used starting from end of Q1, 2024.

⚠ Caution

Additional IP addresses were added to this region.

Region (Region Host)	Hosts	IP Address
Action: If you restrict system access by <i>allowlisting</i> IPs in firewall rules, make sure you update your configuration accordingly. The new IP addresses may be used starting from end of Q1, 2024.	connectivitycertsigning.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12 Additional IP addresses (valid starting from late Q1 2024): 13.209.236.215, 43.201.194.105, 43.202.204.5
	connectivitytunnel.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12 Additional IP addresses (valid starting from late Q1 2024): 13.209.236.215, 43.201.194.105, 43.202.204.5
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)	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
<i>Enterprise & Trial</i>		
Canada (Montreal) - AWS (cf.ca10.hana.ondemand.com)	connectivitynotification.cf.ca10.hana.ondemand.com	3.98.102.153, 35.182.75.101, 35.183.74.34 Additional IP addresses (valid starting from late Q1 2024): 15.157.88.166, 3.98.202.222, 52.60.210.33
	⚠ Caution Additional IP addresses were added to this region. Action: If you restrict system access by <i>allowlisting</i> IPs in firewall rules, make sure you update your config-	

Region (Region Host)	Hosts	IP Address
<div> <div></div> <div>uration accordingly. The new IP addresses may be used starting from end of Q1, 2024.</div> </div>	connectivitycertsigning.cf.ca10.hana.ondemand.com	3.98.102.153, 35.182.75.101, 35.183.74.34 Additional IP addresses (valid starting from late Q1 2024): 15.157.88.166, 3.98.202.222, 52.60.210.33
	connectivitytunnel.cf.ca10.hana.ondemand.com	3.98.102.153, 35.182.75.101, 35.183.74.34 Additional IP addresses (valid starting from late Q1 2024): 15.157.88.166, 3.98.202.222, 52.60.210.33
	connectivitynotification.cf.ch20.hana.ondemand.com	20.208.56.83
	connectivitycertsigning.cf.ch20.hana.ondemand.com	20.208.56.83
Switzerland (Zurich) - Azure (cf.ch20.hana.ondemand.com)	connectivitytunnel.cf.ch20.hana.ondemand.com	20.208.56.83
	connectivitynotification.cf.in30.hana.ondemand.com	34.93.125.74
	connectivitycertsigning.cf.in30.hana.ondemand.com	34.93.125.74
India (Mumbai) - Google Cloud (cf.in30.hana.ondemand.com)	connectivitytunnel.cf.in30.hana.ondemand.com	34.93.125.74
	connectivitynotification.cf.cn40.platform.sapcloud.cn	139.224.7.71
	connectivitycertsigning.cf.cn40.platform.sapcloud.cn	139.224.7.71
China (Shanghai) - Alibaba Cloud (cf.cn40.platform.sapcloud.cn)	connectivitytunnel.cf.cn40.platform.sapcloud.cn	139.224.7.71
Back to Network [page 242] Back to Content [page 239]		
Region (Region Host)	Hosts	IP Address

Region (Region Host)	Hosts	IP Address
ABAP Environment		
<div>  Note </div> <p>To enable the full scenario for the SAP BTP ABAP environment, the DNS names below are needed <i>in addition</i> to the IPs of the Connectivity service mentioned above. If you can only configure allowlists for <i>IP addresses</i> in your firewall, please note that these addresses may change at any time and that you need to validate the values with a DNS service of your choice regularly.</p> <p>For more information, see Regions and API Endpoints for the ABAP Environment.</p>		
Europe (Frankfurt) - AWS	*.abap.eu10.hana.ondemand.com	<dynamic>
Europe (Frankfurt) EU Access - AWS	*.abap.eu11.hana.ondemand.com	<dynamic>
Europe (Netherlands) - Azure	*.abap.eu20.hana.ondemand.com	<dynamic>
US East (VA) - AWS	*.abap.us10.hana.ondemand.com	<dynamic>
US East (VA) - Azure	*.abap.us21.hana.ondemand.com	<dynamic>
US West (WA) - Azure	*.abap.us20.hana.ondemand.com	<dynamic>
Brazil (São Paulo) - AWS	*.abap.br10.hana.ondemand.com	<dynamic>
Japan (Tokyo) - AWS	*.abap.jp10.hana.ondemand.com	<dynamic>
Japan (Tokyo) - Azure	*.abap.jp20.hana.ondemand.com	<dynamic>
Australia (Sydney) - AWS	*.abap.ap10.hana.ondemand.com	<dynamic>
Australia (Sydney) - Azure	*.abap.ap20.hana.ondemand.com	<dynamic>
Asia Pacific (Singapore) - AWS	*.abap.ap11.hana.ondemand.com	<dynamic>
Asia Pacific (Seoul) - AWS	*.abap.ap12.hana.ondemand.com	<dynamic>
Singapore - Azure	*.abap.ap21.hana.ondemand.com	<dynamic>
Canada (Montreal) - AWS	*.abap.ca10.hana.ondemand.com	<dynamic>
Switzerland (Zurich) - Azure	*.abap.ch20.hana.ondemand.com	<dynamic>

Back to [Network \[page 242\]](#)

Back to [Content \[page 239\]](#)


Region (Region Host)	Hosts	IP Address
Neo Environment		
Europe (Rot)	connectivitynotification.hana.ondemand.com	155.56.210.83
(hana.ondemand.com)	connectivitycertsigning.hana.ondemand.com	155.56.210.43
(eu1.hana.ondemand.com)	connectivitytunnel.hana.ondemand.com	155.56.210.84
Europe (Frankfurt)	connectivitynotification.eu2.hana.ondemand.com	157.133.206.143

Region (Region Host)	Hosts	IP Address
(eu2.hana.ondemand.com)	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
	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

Region (Region Host)	Hosts	IP Address
Canada (Toronto) (ca1.hana.ondemand.com)	connectivitynotification.ca1.hana.ondemand.com	130.214.174.201
	connectivitycertsigning.ca1.hana.ondemand.com	130.214.174.220
	connectivitytunnel.ca1.hana.ondemand.com	130.214.174.236
Brazil (São Paulo) (br1.hana.ondemand.com)	connectivitynotification.br1.hana.ondemand.com	130.214.96.193
	connectivitycertsigning.br1.hana.ondemand.com	130.214.96.195
	connectivitytunnel.br1.hana.ondemand.com	130.214.96.173
UAE (Dubai) (ae1.hana.ondemand.com)	connectivitynotification.ae1.hana.ondemand.com	130.214.80.206
	connectivitycertsigning.ae1.hana.ondemand.com	130.214.80.208
	connectivitytunnel.ae1.hana.ondemand.com	130.214.80.182
KSA (Riyadh) (sa1.hana.ondemand.com)	connectivitynotification.sa1.hana.ondemand.com	130.214.209.241
	connectivitycertsigning.sa1.hana.ondemand.com	130.214.209.207
	connectivitytunnel.sa1.hana.ondemand.com	130.214.209.191

Back to [Network \[page 242\]](#)

Back to [Content \[page 239\]](#)

Region (Region Host)	Hosts	IP Address
Trial (Cloud Foundry Environment)		
<div>  Note </div> <p>In the Cloud Foundry environment, IPs are controlled by the respective IaaS provider (AWS, Azure, or Google Cloud). IPs may change due to network updates on the provider side. Any planned changes will be announced several weeks before they take effect. See also Regions.</p>		
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

Region (Region Host)	Hosts	IP Address
United States East (VA) - AWS (cf.us10.hana.ondemand.com)	connectivitynotification.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211
	connectivitycertsigning.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211
	connectivitytunnel.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211
Singapore - Azure (cf.ap21.hana.ondemand.com)	connectivitynotification.cf.ap21.hana.ondemand.com	20.184.61.122
	connectivitycertsigning.cf.ap21.hana.ondemand.com	20.184.61.122
	connectivitytunnel.cf.ap21.hana.ondemand.com	20.184.61.122

Back to [Network \[page 242\]](#)

Back to [Content \[page 239\]](#)

Note

If you install the Cloud Connector in a network segment that is isolated from the backend systems, make sure the exposed hosts and ports are still reachable and open them in the firewall that protects them:

- for HTTP, the ports you chose for the HTTP/S server.
- for LDAP, the port of the LDAP server.
- for RFC, it depends on whether you use an SAProuter or not and whether load balancing is used:
 - if you use an SAProuter, it is typically configured as visible in the network of the Cloud Connector and the corresponding *routtab* is exposing all the systems that should be used.
 - without SAProuter, you must open the application server hosts and the corresponding gateway ports (33##, 48##). When using load balancing for the connection, you must also open the message server host and port.

See also [Network Zones \[page 256\]](#).

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

Back to [Network \[page 242\]](#)

Back to [Content \[page 239\]](#)

Related Information

[Installation on Microsoft Windows OS \[page 265\]](#)

[Installation on Linux OS \[page 268\]](#)

[Installation on Mac OS X \[page 271\]](#)

[Recommendations for Secure Setup \[page 272\]](#)

[Sizing Recommendations \[page 258\]](#)

1.5.1.1.1 Network Zones

Choose a network zone for your Cloud Connector installation.

A customer network is usually divided into multiple network zones or subnetworks according to the security level of the contained components. For example, the DMZ that contains and exposes the external-facing services of an organization to an untrusted network, usually the Internet, and there are one or more other network zones which contain the components and services provided in the company's intranet.

You can generally choose the network zone in which to set up the Cloud Connector:

- Internet access to the SAP BTP region host, either directly or via HTTPS proxy.
- Direct access to the internal systems it provides access to, which means that there is transparent connectivity between the Cloud Connector and the internal system.

The Cloud Connector can be set up either in the DMZ and operated centrally by the IT department, or set up in the intranet and operated by the appropriate line of business.

Note

The internal network must allow access to the required ports; the specific configuration depends on the firewall software used.

The default ports are **80** for HTTP and **443** for HTTPS. For RFC communication, you need to open a gateway port (default: **33+<instance number>**) and an arbitrary message server port. For a connection to a HANA Database (on SAP BTP) via JDBC, you need to open an arbitrary *outbound* port in your network. Mail (SMTP) communication is not supported.

1.5.1.1.2 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 `<ccc_dir>/log/` within the Cloud Connector `root` directory. The `ljs_trace.log` file contains traces in general, communication payload traces are stored in `traffic_trace_*.trc`. These files may be used by SAP Support to analyze potential issues. The default trace level is `Information`, where the amount of written data is generally only a few KB per day. You can turn off these traces to save disk space. However, we recommend that you don't turn off this trace completely, but that you leave it at the default settings, to allow root cause analysis if an issue occurs. If you set the trace level to `All`, the amount of data can easily reach the range of several GB per day. Use trace level `All` only to analyze a specific issue. Payload trace, however, should normally be turned off, and used only for analysis by SAP Support.

Note

Regularly back up or delete written trace files to clean up the used disk space.

Audit log files are written to `/log/audit/<subaccount-name>/audit-log_<subaccount-name>_<date>.csv` within the Cloud Connector root directory. By default, only security-related events are written in the audit log. You can change the audit log level using the administration UI, as described in [Manage Audit Logs \[page 576\]](#).

To be compliant with the regulatory requirements of your organization and the regional laws, the audit log files must be persisted for a certain period of time for traceability purposes. Therefore, we recommend that you back up the audit log files regularly from the Cloud Connector file system and keep the backup for the length of time required.

Related Information

[Prerequisites \[page 239\]](#)

1.5.1.2 Sizing Recommendations

When installing a Cloud Connector, the first thing you need to decide is the sizing of the installation.

This section gives some basic guidance what to consider for this decision. The provided information includes the shadow instance, which should always be added in productive setups. See also [Install a Failover Instance for High Availability \[page 538\]](#).

📘 Note

The following recommendations are based on current experiences. However, they are only a rule of thumb since the actual performance strongly depends on the specific environment. The overall performance of a Cloud Connector is impacted by many factors (number of hosted subaccounts, bandwidth, latency to the attached regions, network routers in the corporate network, used JVM, and others).

Restrictions

The sizing data refer to a single Cloud Connector installation.

📘 Note

Up until now, you cannot perform horizontal scaling directly. However, you can distribute the load statically by operating multiple Cloud Connector installations with different location IDs for all involved subaccounts. In this scenario, you can use multiple destinations with virtually the same configuration, except for the location ID. See also [Managing Subaccounts \[page 291\]](#), step 4. Alternatively, each of the Cloud Connector instances can host its own list of subaccounts without any overlap in the respective lists. Thus, you can handle more load, if a single installation risks to be overloaded.

Related Information

[Hardware Setup \[page 258\]](#)

[Configuration Setup \[page 262\]](#)

1.5.1.2.1 Hardware Setup

How to choose the right sizing for your Cloud Connector installation.

Regarding the hardware, we recommend that you use different setups for master and shadow. One dedicated machine should be used for the master, another one for the shadow. Usually, a shadow instance takes over the master role only temporarily. During most of its lifetime, in the shadow state, it needs less resources compared to the master.

If the master instance is available again after a downtime, we recommend that you switch back to the actual master.

Note

The sizing recommendations refer to the overall load across all subaccounts that are connected via the Cloud Connector. This means that you need to accumulate the expected load of all subaccounts and should not only calculate separately per subaccount (taking the one with the highest load as basis).

Related Information

[Sizing for the Master Instance \[page 259\]](#)

[Sizing for the Shadow Instance \[page 261\]](#)

1.5.1.2.1.1 Sizing for the Master Instance

Learn more about the basic criteria for the sizing of your Cloud Connector master instance.

For the master setup, keep in mind the expected load for communication between the SAP BTP and on-premise systems. The setups listed below differ in a mostly qualitative manner, without hard limits for each of them.

Note

The mentioned sizes are considered as minimal configuration, larger ones are always ok. In general, the more applications, application instances, and subaccounts are connected, the more competition will exist for the limited resources on the machine.

Installation Size (S, M, L)	CPU (x86_64)	Machine Memory / Heap / Direct Memory	Disk Space
S: The expected load is small (up to 1 million requests per day, request concurrency and size is in average low) and only a few subaccounts with some applications are connected. In addition, only a few serv- ice channels are used, only small data amount is repli- cated to cloud systems. Setup can be a done in a vir- tual or physical machine.	2 cores 2.6 GHz	4GB RAM / 1GB / 2GB	10GB
M: The expected load is medium (up to 10 million requests per day, request concurrency and size is in average me- dium) and multiple subac- counts with multiple appli- cations are connected. In addition, many service channels are used, and a medium data amount is re- plicated to cloud systems. We recommend that you do the setup in a virtual or physical machine. A virtual machine should be on a host without overcommitted resources.	4 cores 3.0 Ghz	16GB RAM / 4GB / 8GB	20GB

Installation Size (S, M, L)	CPU (x86_64)	Machine Memory / Heap / Direct Memory	Disk Space
L:	8 cores 3.0 Ghz	32GB RAM / 8GB / 16GB	40GB
<p>The expected load is large (more than 10 million requests per day, request concurrency and size is in average medium or high) and multiple subaccounts with multiple applications are connected.</p> <p>In addition, many service channels are used, and a large data amount is replicated 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>			

Particularly the **heap size** is critical. If you size it too low for the load passing the Cloud Connector, at some point the Java Virtual Machine will execute full GCs (garbage collections) more frequently, blocking the processing of the Cloud Connector completely for multiple seconds, which massively slows down overall performance. If you experience such situations regularly, you should increase the heap size in the Cloud Connector UI (choose ► [Configuration](#) ► [Advanced](#) ► [JVM](#) ►). See also [Configure the Java VM \[page 510\]](#).

📘 Note

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

1.5.1.2.1.2 Sizing for the Shadow Instance

Learn more about the basic criteria for the sizing of your Cloud Connector shadow instance (high availability mode).

The shadow installation is typically not used in standard situations and hence does not need the same sizing, assuming that the time span in which it takes over the master role is limited.

📘 Note

The shadow only acts as master, for example, during an upgrade or when an abnormal situation occurs on the master machine, and either the Cloud Connector or the full machine on OS level needs to be restarted.

While being in the shadow state, the resource consumption is very low, especially in productive environments, where typically only few configuration changes are required. Therefore, the machine sizing can usually be smaller than the one for the master. However, if you want to mitigate the risk of a longer outage of the master machine, you should increase the sizing of the shadow up to the master size:

Master	Shadow
S	S installation as virtual machine
M	S installation (with double memory) as virtual or physical machine M installation as virtual or physical machine for risk mitigation
L	M installation as virtual or physical machine L installation as virtual or physical machine for risk mitigation

1.5.1.2.2 Configuration Setup

Choose the right connection configuration options to improve the performance of the Cloud Connector.

This section provides detailed information how you can adjust the configuration to improve overall performance. This is typically relevant for an M or L installation (see [Hardware Setup \[page 258\]](#)). 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 \[page 507\]](#).
- In versions prior to 2.11, you have to modify the configuration files with an editor and restart the Cloud Connector to activate the changes.

In general, the Cloud Connector tunnel is multiplexing multiple virtual connections over a single physical connection. Thus, a single connection can handle a considerable amount of traffic. However, increasing the maximum number of physical connections allows you to make use of the full available bandwidth and to minimize latency effects.

If the bandwidth limit of your network is reached, adding additional connections doesn't increase the throughput, but will only consume more resources.

Note

Different network access parameters may impact and limit your configuration options: if the access to an external network is a 1 MB line with an added latency of 50 ms, you will not be able to achieve the same data volumes like with a 10 GB line with an added latency of < 1 ms. However, even if the line is good, for example 10 GB, but with an added latency of 100 ms, the performance might still be bad.

Optimal configuration strongly depends on your actual scenarios. A good approach is to try out different settings, if the current performance does not meet your expectations.

Related Information

[On-Demand To On-Premise Connections \[page 263\]](#)

[On-Premise To On-Demand Connections \(Service Channels\) \[page 264\]](#)

1.5.1.2.2.1 On-Demand To On-Premise Connections

Configure the physical connections for on-demand to on-premise calls in the Cloud Connector.

Adjusting the number of physical connections for this direction is possible both globally in the Cloud Connector UI (► [Configuration](#) ► [Advanced](#) ►), and for individual communication partners on cloud side (► [On-Demand To On-Premise](#) ► [Applications](#) ►). For an application/instance in the SAP BTP Neo environment, you can define settings even per Java application or HANA instance.

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](#) ►. For Neo applications with multiple processes, the configured connections are established per process, which lets you use lower overall values for such a connection.

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, if no specific value is set (► [On-Demand To On-Premise](#) ► [Applications](#) ►).

To set application-specific values, choose ► [Subaccount](#) ► [Cloud To On-Premise](#) ►, tab [Applications](#), section [Tunnel Connection Limits](#). Here you can define the number of physical connections per application.

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, instead of specifying individual values per application.

The following simple rule helps you to decide, whether an individual setting for a concrete application is required:

- Per 20 threads in one process executing requests to on-premise systems, provide 1 physical connection.
- If the request or response net size is larger than 250k, make sure to add an additional connection per 2 of such clients.

📘 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.

Example

For an application in the SAP BTP Neo environment, requests to on-premise systems are executed in each application thread. The expected usage is 100 concurrent users. In average, about 3 of those users are triggering a remote call to an on-premise system that returns about 400k. That is, for the number of threads, you should use 5 physical connections, for the 3 clients sending larger amounts add an additional 2, which sums up to 7 connections.

Tunnel Worker Threads

In addition to the number of connections, you can configure the number of `<Tunnel Worker Threads>`. This value should be at least equal to the maximum of all individual application tunnel connections in all subaccounts, to have at least 1 thread available for each connection that can process incoming requests and outgoing responses.

Protocol Processor Worker Threads

The value for `<Protocol Processor Worker Threads>` is mainly relevant if RFC is used as protocol. Since its communication model towards the ABAP system is a blocking one, each thread can handle only one call at a time and cannot be shared. Hence, you should provide 1 thread per 5 concurrent RFC requests.

Note

The longer the RFC execution time in the backend, the more threads you should provide. Threads can be reused only after the response of a call was returned to SAP BTP.

1.5.1.2.2.2 On-Premise To On-Demand Connections (Service Channels)

Configure the number of physical connections for a Cloud Connector service channel.

Service channels let you configure the number of physical connections to the communication partner on cloud side, see [Using Service Channels \[page 491\]](#). The default is 1. This value is used as well in versions prior to Cloud Connector 2.11, which did not offer a configuration option for each service channel. You should define the number of connections depending on the expected number of clients and, with lower priority, depending on the size of the exchanged messages.

If there is only a single RFC client for an S/4HANA Cloud channel or only a single HANA client for a HANA DB on SAP BTP side, increasing the number doesn't help, as each virtual connection is assigned to one physical connection. The following simple rule lets you to define the required number of connections per service channel:

- Per 10 concurrent clients, use one physical connection.

- If the transferred net data size is larger than 500k per request, make sure to add an additional connection per 2 of such clients.

Example

For a HANA system in the SAP BTP, data is replicated using 18 concurrent clients in the on-premise network. In average, about 5 of those clients are regularly sending 600k. For the number of clients, you should use 2 physical connections, for the 5 clients sending larger amounts add an additional 3, which sums up to 5 connections.

1.5.1.3 Installation on Microsoft Windows OS

Installing the Cloud Connector on a Microsoft Windows operating system.

Context

You can choose between a simple `portable` variant of the Cloud Connector and the MSI-based `installer`. The `installer` is the generally recommended version that you can use for both developer and productive scenarios. It lets you, for example, register the Cloud Connector as a Windows service and this way automatically start it after machine reboot.

→ Tip

If you are a developer, you might want to use the `portable` variant as you can run the Cloud Connector after a simple unzip (archive extraction). You might want to use it also if you cannot perform a full installation due to lack of permissions, or if you want to use multiple versions of the Cloud Connector simultaneously on the same machine.

Prerequisites

- You have one of the supported 64-bit operating systems. For more information, see [Product Availability Matrix \[page 241\]](#).
- You have downloaded either the `portable` variant as ZIP archive for Windows, or the MSI `installer` from the [SAP Development Tools for Eclipse](#) page.
- You must install Microsoft Visual Studio C++ 2013 *and* (for the latest SAP JVM versions) Microsoft Visual Studio C++ 2019 runtime libraries (download `vc_redist_x64.exe` for each version). For more information, see [Visual C++ Redistributable Packages for Visual Studio 2013](#) 📄 and [Microsoft Visual C++ Redistributable latest supported downloads](#) 📄.

Note

Even with the more recent version 2019, you still must install the Microsoft Visual Studio C++ 2013 libraries, since they are needed for the Cloud Connector.

- A supported Java version must be installed. For more information, see [JDKs \[page 241\]](#). If you want to use SAP JVM, you can download it from the [SAP Development Tools for Eclipse](#) page.
- When using the `portable` variant, the environment variable `<JAVA_HOME>` must be set to the Java installation directory, so that the `bin` subdirectory can be found. Alternatively, you can add the relevant `bin` subdirectory to the `<PATH>` variable.

Portable Scenario

1. Extract the `<sapcc-<version>-windows-x64.zip>` ZIP file to an arbitrary directory on your local file system.
2. Set the environment variable `JAVA_HOME` to the installation directory of the JDK that you want to use to run the Cloud Connector. Alternatively, you can add the `bin` subdirectory of the JDK installation directory to the `PATH` environment variable.
3. Go to the Cloud Connector installation directory and start it using the `go.bat` batch file.
4. Continue with the **Next Steps** section.

Note

The Cloud Connector is not started as a service when using the portable variant, and hence will not automatically start after a reboot of your system. Also, the portable version does not support the automatic upgrade procedure. `sapcc-<version>-windows-x64.msi`

Installer Scenario

1. Start the `<>sapcc-` installer by double-clicking it.
2. The installer informs you that you are now guided through the installation process. Choose **Next>**.
3. Navigate to the desired installation directory for your Cloud Connector and choose **Next>**. When doing the installation in the context of an upgrade, make sure you choose the previous installation directory again.
4. You can choose the port on which the administration UI is reachable. Either leave the default **8443** or choose a different port if needed. Then, choose **Next>**.
5. Select the JDK to be used for running the Cloud Connector. The installer displays a list of all usable JDKs that are installed on your machine. If the needed JDK is not listed in the drop-down box (for example, if it's an SAP JVM that is not registered in the Windows registry upon installation), you can browse to its installation directory and select it. We recommend that you use an up-to-date Java **8** installation to run the Cloud Connector.
6. Decide whether the Cloud Connector should be started immediately after finishing the setup. Then, choose **Next>**.
7. To start the installation, press the **Next>** button again.

8. After successful installation, choose [Close](#).
9. Continue with the [Next Steps](#) section.

Note

The Cloud Connector is started as a Windows service in the productive use case. Therefore, installation requires administration permissions. After installation, manage this service under [Control Panel](#) [Administrative Tools](#) [Services](#). The service name is `Cloud_Connector` (formerly named `Cloud_Connector_2.0`). Make sure the service is executed with a user that has limited privileges. Typically, privileges allowed for service users are defined by your company policy. Adjust the folder and file permissions to be manageable by only this user and system administrators.

On Windows, the file `scc_service.log` is created and used by the Microsoft MSI installer (during Cloud Connector installation), and by the `scchost.exe` executable, which registers and runs the Windows service if you install the Cloud Connector as a Windows background job.

This log file is only needed if a problem occurs during Cloud Connector installation, or during creation and start of the Windows service, in which the Cloud Connector is running. You can find the file in the `log` folder of your Cloud Connector installation directory.

Starting the Cloud Connector

After installation, the Cloud Connector is registered as a Windows service that is configured to be started automatically after a system reboot. You can start and stop the service via shortcuts on the desktop ("Start Cloud Connector" and "Stop Cloud Connector"), or by using the `Windows Services` manager and look for the service **SAP Cloud Connector**.

Access the Cloud Connector administration UI at <https://localhost:<port>>, where the default port is **8443** (but this port might have been modified during the installation).

Next Steps

1. Open a browser and enter: <https://<hostname>:8443>. `<hostname>` is the host name of the machine on which you have installed the Cloud Connector. If you access the Cloud Connector locally from the same machine, you can simply enter `localhost`.
2. Continue with the initial configuration of the Cloud Connector, see [Initial Configuration \[page 278\]](#).

Related Information

[\(Optional\) Install SAP JVM](#)

[Recommendations for Secure Setup \[page 272\]](#)

1.5.1.4 Installation on Linux OS

Installing the Cloud Connector on a Linux operating system.

Context

You can choose between a simple `portable` variant of the Cloud Connector and the RPM-based `installer`. The `installer` is the generally recommended version that you can use for both the developer and the productive scenario. It registers, for example, the Cloud Connector as a daemon service and this way automatically starts it after machine reboot.

→ Tip

If you are a developer, you might want to use the `portable` variant as you can run the Cloud Connector after a simple `"tar -xzf"` execution. You also might want to use it if you cannot perform a full installation due to missing permissions for the operating system, or if you want to use multiple versions of the Cloud Connector simultaneously on the same machine.

Prerequisites

- You have one of the supported 64-bit operating systems. For more information, see [Product Availability Matrix \[page 241\]](#).
- The supported platforms are `x64` and `ppc64le`, represented below by the variable `<platform>`. Variable `<arch>` is `x86_64` or `ppc64le` respectively.
- You have downloaded either the `portable` variant as `tar.gz` archive for Linux or the RPM `installer` contained in the ZIP for Linux, from [SAP Development Tools for Eclipse](#).
- A supported Java version must be installed. For more information, see [JDKs \[page 241\]](#).
If you want to use SAP JVM, you can download it from the [SAP Development Tools for Eclipse](#) page. Use the following command to install it:

```
rpm -i sapjvm-<version>-linux-<platform>.rpm
```

If you want to check the JVM version installed on your system, use the following command:

```
rpm -qa | grep jvm
```

When installing it using the RPM package, the Cloud Connector will detect it and use it for its runtime.

- When using the `tar.gz` archive, the environment variable `<JAVA_HOME>` must be set to the Java installation directory, so that the `bin` subdirectory can be found. Alternatively, you can add the Java installation's `bin` subdirectory to the `<PATH>` variable.

Portable Scenario

1. Extract the *tar.gz* file to an arbitrary directory on your local file system using the following command:

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

Note

If you use the parameter "o", the extracted files are assigned to the user ID and the group ID of the user who has unpacked the archive. This is the default behavior for users other than the `root` user.

2. Go to this directory and start the Cloud Connector using the `go.sh` script.
3. Continue with the **Next Steps** section.

Note

In this case, the Cloud Connector is not started as a daemon, and therefore will not automatically start after a reboot of your system. Also, the `portable` version does not support the automatic upgrade procedure.

Installer Scenario

1. Extract the *sapcc-<version>-linux-<platform>.zip* archive to an arbitrary directory by using the following the command:

```
unzip sapcc-<version>-linux-<platform>.zip
```

2. Go to this directory and install the extracted RPM using the following command. You can perform this step only as a `root` user.

```
rpm -i com.sap.scc-ui-<version>.<arch>.rpm
```

3. Continue with the **Next Steps** section.

In the productive case, the Cloud Connector is started as a daemon. If you need to manage the daemon process, execute:

```
System V init distributions: service scc_daemon stop|restart|start|status  
systemd distributions: systemctl stop|restart|start|status scc_daemon
```

Caution

When adjusting the Cloud Connector installation (for example, restoring a backup), make sure the RPM package management is synchronized with such changes. If you simply replace files that do not fit to the information stored in the package management, lifecycle operations (such as upgrade or uninstallation) might fail with errors. Also, the Cloud Connector might get into unrecoverable state.

Example: After a file system restore, the system files represent Cloud Connector 2.3.0 but the RPM package management "believes" that version 2.4.3 is installed. In this case, commands like `rpm -U` and `rpm -e` do not work as expected. Furthermore, avoid using the `--force` parameter as it may lead to an unpredictable state with two versions being installed concurrently, which is not supported.

Extending the Daemon (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.

Note

Make sure `JAVA_HOME` is set to the JVM used (in the shell where the command is executed).

Execute the following command:

```
System V init distributions: /opt/sap/scc/daemon.sh reinstall
systemd distributions: /opt/sap/scc/daemon.sh reinstallSystemd
```

The daemon extension will survive Cloud Connector version updates.

Starting the Cloud Connector

After installation via RPM manager, the Cloud Connector process is started automatically and registered as a daemon process, which ensures the automatic restart of the Cloud Connector after a system reboot.

To start, stop, or restart the process explicitly, open a command shell and use the following commands, which require `root` permissions:

```
System V init distributions: service scc_daemon start|stop|restart
systemd distributions: systemctl start|stop|restart scc_daemon
```

Next Steps

1. Open a browser and enter: <https://<hostname>:8443>. `<hostname>` is the host name of the machine on which you installed the Cloud Connector.
If you access the Cloud Connector locally from the same machine, you can simply enter `localhost`.
2. Continue with the initial configuration of the Cloud Connector, see [Initial Configuration \[page 278\]](#).

Related Information

[Recommendations for Secure Setup \[page 272\]](#)

1.5.1.5 Installation on Mac OS X

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

Prerequisites

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 \[page 241\]](#).
- 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 \[page 241\]](#). 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 -xzf 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.

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 \[page 278\]](#).

Related Information

[Recommendations for Secure Setup \[page 272\]](#)

1.5.1.6 Recommendations for Secure Setup

For the Connectivity service and the Cloud Connector, you should apply the following guidelines to guarantee the highest level of security for these components.

Security Status

From the [Connector](#) menu, choose [Security Status](#) to access an overview showing potential security risks and the recommended actions.

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar contains a navigation menu with 'Connector' selected, and a sub-menu for 'Security Status' which includes 'Alerting', 'High Availability', 'Hardware Metrics Monitor', and 'Configuration'. The main content area is titled 'Security Status' and shows a 'Risk' indicator. Below this, there are two sections: 'General Security Status' and 'Subaccount-Specific Security Status'.

General Security Status

Status	Area	Description	Actions
⚠	UI Certificate	Replace the default UI certificate with a certificate that uses the host name as its common name (CN)	>
⚠	Cipher Suites	Only TLS ciphers with SHA256 (or greater bit lengths) are deemed secure	>
⚠	Trust Store	Trust store is empty — no access restrictions	>
⚠	Authentication	Configure local LDAP for authentication of Cloud Connector users	>
✅	CPIC Trace	Trace is off	>
⚠	Service User	Set up service user specifically for this Cloud Connector	✎

Subaccount-Specific Security Status

Display Name	Application White-List	Payload Trace
conn2	⚠ White-list is empty — all applications will be trusted	✅ Trace is off
conn3	⚠ White-list is empty — all applications will be trusted	✅ Trace is off

The *General Security Status* addresses security topics that are subaccount-independent.

- Choose any of the *Actions* icons in the corresponding line to navigate to the UI area that deals with that particular topic and view or edit details.

Note

Navigation is not possible for the last item in the list (*Service User*).

- The service user is specific to the Windows operating system (see [Installation on Microsoft Windows OS \[page 265\]](#) for details) and is only visible when running the Cloud Connector on Windows. It cannot be accessed or edited through the UI. If the service user was set up properly, choose *Edit* and check the corresponding checkbox.

The *Subaccount-Specific Security Status* lists security-related information for each and every subaccount.

Note

The security status only serves as a reminder to address security issues and shows if your installation complies with all recommended security settings.

UI Access

Upon installation, the Cloud Connector provides an initial user name and password for the administration UI, and forces the user (**Administrator**) to change the password. You must change the password immediately after installation.

The connector itself does not check the strength of the password. You should select a strong password that cannot be guessed easily.

Note

To enforce your company's password policy, we recommend that you configure the Administration UI to use an LDAP server for authorizing access to the UI.

The default user store is a local file store. It allows only one user, and only the *Administrator* role for this user. Using an LDAP server as user store lets you create various users to access the UI, and assign different roles to them. For more information on available roles, see [Use LDAP for User Administration \[page 521\]](#).

The Cloud Connector administration UI can be accessed remotely via HTTPS. The connector uses a standard X.509 self-signed certificate as SSL server certificate. You can exchange this certificate with a specific certificate that is trusted by your company. See [Exchange UI Certificates in the Administration UI \[page 517\]](#).

Note

Since browsers usually do not resolve *localhost* to the host name whereas the certificate usually is created under the host name, you might get a certificate warning. In this case, simply skip the warning message.

OS-Level Access

The Cloud Connector is a security-critical component that handles the external access to systems of an isolated network, comparable to a reverse proxy. We therefore recommend that you restrict the access to the operating system on which the Cloud Connector is installed to the minimal set of users who would administrate the Cloud Connector. This minimizes the risk of unauthorized users getting access to credentials, such as certificates stored in the secure storage of the Cloud Connector.

We also recommend that you use the machine to operate only the Cloud Connector and no other systems.

Administrator Privileges

To log on to the Cloud Connector administration UI, the **Administrator** user of the connector must not have an operating system (OS) user for the machine on which the connector is running. This allows the OS administrator to be distinguished from the Cloud Connector administrator. To make an initial connection between the connector and a particular SAP BTP subaccount, you need an SAP BTP user with the required permissions for the related subaccount. We recommend that you separate these roles/duties (that means, you have separate users for Cloud Connector administrator and SAP BTP).

Note

We recommend that only a small number of users are granted access to the machine as *root* users.

Hard Drive Encryption

Hard drive encryption for machines with a Cloud Connector installation ensures that the Cloud Connector configuration data cannot be read by unauthorized users, even if they obtain access to the hard drive.

Supported Protocols

Currently, the protocols HTTP, HTTPS, RFC, RFC with SNC, LDAP, LDAPS, TCP, and TCP over TLS are supported for connections between the SAP BTP and on-premise systems when the Cloud Connector and the Connectivity service are used. The whole route from the application virtual machine in the cloud to the Cloud Connector is always SSL-encrypted.

The route from the connector to the back-end system can be TLS-encrypted or SNC-encrypted. See [Configure Access Control \(HTTP\) \[page 342\]](#) and [Configure Access Control \(RFC\) \[page 350\]](#).

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](#).

Connector

Security Status

Alerting

High Availability

Hardware Metrics Monitor

Configuration

Define Subaccount

Cloud To On-Premise

On-Premise To Cloud

Monitor (Cloud to On-Premise)

Monitor (On-Premise to Cloud)

Audits

Log And Trace Files

Important Links

Legal Information

Configuration

USER INTERFACE

CLOUD

ON PREMISE

REPORTING

ADVANCED

Authentication Mode: Password

User Name: Administrator

UI Certificate

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

Valid To: 2024-12-12 18:55:27 +0100

Subject Alternative Names

Type	Value
No data	

Certificates Chain

Subject DN

CN=SCC,OU=Connectivity,O=SAP SE,C=DE

Cipher Suites (45)

Status Quo	Status New	Security	Name	Actions
◇	□	✓	TLS_AES_128_GCM_SHA256	ⓘ
◇	□	✓	TLS_AES_256_GCM_SHA384	ⓘ
△		⚠	TLS_DHE_DSS_WITH_AES_128_CBC_SHA	ⓘ
△		⚠	TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	ⓘ
△	⊗	✓	TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	ⓘ
△		⚠	TLS_DHE_DSS_WITH_AES_256_CBC_SHA	ⓘ
△		⚠	TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	ⓘ
△		✓	TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	ⓘ
□		⚠	TLS_DHE_RSA_WITH_AES_128_CBC_SHA	ⓘ
□		⚠	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	ⓘ

The first column labeled **Status Quo** shows the current state of all available ciphers. The second column **Status New** shows the state the ciphers will have after a restart, if that state differs from the current one (that is, there is no entry in that column if the state remains the same after a restart).

Ciphers are either enabled or disabled, and they are either compatible or incompatible with the current UI certificate (that is, can potentially be used or not be used) . Consult the tooltips of the (four types of) icons for details. The third column shows SAP's security assessment of the ciphers *as per the time of release*. Enable or disable individual ciphers using the button in the **Action** column. Enable or disable certain groups of ciphers using the appropriate table button. Consult the tooltips for details.

Note

We recommend that you enable all cipher suites that are compatible with the UI whenever you plan to switch to another JVM. You can comfortably do so by using the first button to the right of the filter buttons.

As the set of supported ciphers may differ, the selected ciphers may not be supported by the new JVM. In that case, the Cloud Connector will not start anymore, and you must fix the issue by manually adapting the file `conf/server.xml`. After a successful switch, you can adjust the list of eligible ciphers again.

Related Information

[Connectivity via Reverse Proxy \[page 614\]](#)

[Security \[page 589\]](#)

1.5.2 Configuration

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

Topic	Description
Initial Configuration [page 278]	After installing the Cloud Connector and starting the Cloud Connector daemon, you can log on and perform the required configuration to make your Cloud Connector operational.
Managing Subaccounts [page 291]	How to connect SAP BTP subaccounts to your Cloud Connector.
Authenticating Users against On-Premise Systems [page 303]	Basic authentication and principal propagation (user propagation) are the authentication types currently supported by the Cloud Connector.
Configure Access Control [page 340]	Configure access control or copy the complete access control settings from another subaccount on the same Cloud Connector.
Configuration REST APIs [page 367]	Configure a newly installed Cloud Connector (initial configuration, subaccounts, access control) using the configuration REST API.
Configure an On-Premise User Store [page 489]	Configure applications running on SAP BTP to use your corporate LDAP server as a user store.
Using Service Channels [page 491]	Service channels provide access from an external network to certain services on SAP BTP, which are not exposed to direct access from the Internet.
Configure Trust [page 501]	Set up an allowlist for trusted cloud applications and a trust store for on-premise systems in the Cloud Connector.
Connect DB Tools to SAP HANA via Service Channels [page 495]	How to connect database, BI, or replication tools running in the on-premise network to a HANA database on SAP BTP using the service channels of the Cloud Connector.
Configure Domain Mappings for Cookies [page 504]	Map virtual and internal domains to ensure correct handling of cookies in client/server communication.
Configure Solution Management Integration [page 506]	Activate Solution Management reporting in the Cloud Connector.
Configure Advanced Connectivity [page 507]	Adapt connectivity settings that control the throughput and HTTP connectivity to on-premise systems.
Configure the Java VM [page 510]	Adapt the JVM settings that control memory management.
Configuration Backup [page 511]	Backup and restore your Cloud Connector configuration.
Configure Login Screen Information [page 512]	Add additional information to the login screen and configure its appearance.

1.5.2.1 Initial Configuration

After installing and starting the Cloud Connector, log on to the administration UI and perform the required configuration to make your Cloud Connector operational.

Tasks

[Prerequisites \[page 278\]](#)

[Log on to the Cloud Connector \[page 279\]](#)

[Initial Setup \[page 280\]](#)

[Set up Connection Parameters and HTTPS Proxy \[page 281\]](#)

[Establish Connections to SAP BTP \[page 284\]](#)

Prerequisites

- You have downloaded and installed the Cloud Connector, see [Installation \[page 238\]](#).
- You have assigned one of these roles/role collections to the subaccount user that you use for initial Cloud Connector setup, depending on the SAP BTP environment in which your subaccount is running:

Note

For the **Cloud Foundry** environment, you must know on which cloud management tools feature set (A or B) your account is running. For more information on feature sets, see [Cloud Management Tools — Feature Set Overview](#).

Environment	Required Roles/Role Collections	More Information
Cloud Foundry [feature set A]	<p>The user must be a <i>member of the global account</i> that the subaccount belongs to.</p> <p>Alternatively, you can assign the user as <i>Security Administrator</i>.</p>	<p>Add Members to Your Global Account</p> <p>Managing Security Administrators in Your Subaccount [Feature Set A]</p>

Environment	Required Roles/Role Collections	More Information
Cloud Foundry [feature set B]	<p>Assign at least one of these <i>default role collections</i> (all of them including the role Cloud Connector Administrator):</p> <ul style="list-style-type: none"> • Subaccount Administrator • Cloud Connector Administrator • Connectivity and Destination Administrator <p>Alternatively, you can assign a <i>custom role collection</i> to the user that includes the role Cloud Connector Administrator.</p>	Role Collections and Roles in Global Accounts, Directories, and Subaccounts [Feature Set B]
Neo	<p>Assign at least one of these <i>default roles</i>:</p> <ul style="list-style-type: none"> • Cloud Connector Admin • Administrator <p>Alternatively, you can assign a <i>custom role</i> to the user that includes the permission <code>manageSCCTunnels</code>.</p>	Managing Member Authorizations in the Neo Environment

After establishing the Cloud Connector connection, this user is not needed any more, since it serves only for initial connection setup. You may revoke the corresponding role assignment then and remove the user from the [Members](#) list (**Neo** environment), or from the [Users](#) list (**Cloud Foundry** environment).

Note

If the Cloud Connector is installed in an environment that is operated by SAP, SAP provides a user that you can add as member in your SAP BTP subaccount and assign the required role.

- We strongly recommend that you read and follow the steps described in [Recommendations for Secure Setup \[page 272\]](#). For operating the Cloud Connector securely, see also [Security Guidelines \[page 596\]](#).

Back to [Tasks \[page 278\]](#)

Log on to the Cloud Connector

To administer the Cloud Connector, you need a Web browser. To check the list of supported browsers, see [Prerequisites and Restrictions](#) → section *Browser Support*.

1. In a Web browser, enter: **https://<hostname>:<port>**
 - **<hostname>** refers to the machine on which the Cloud Connector is installed. If installed on your machine, you can simply enter **localhost**.
 - **<port>** is the Cloud Connector port specified during installation (the default port is **8443**).
2. On the logon screen, enter **Administrator / manage** (case sensitive) for **<User Name> / <Password>**.

Note

By default, the Cloud Connector includes a self-signed UI certificate. Browsers may show a security warning because they don't trust the issuer of this certificate. In this case, you can skip the warning message.

Back to [Tasks \[page 278\]](#)

Initial Setup

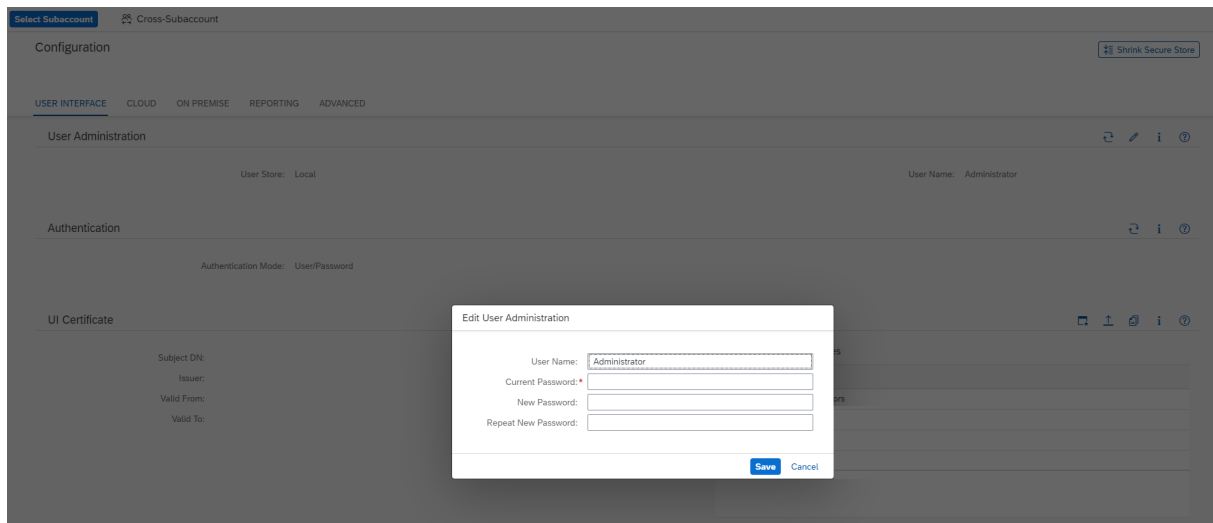
1. When you first log in, you must change the password before you continue, regardless of the installation type you have chosen.
2. Choose between master and shadow installation. Use **Master** if you are installing a single Cloud Connector instance or a main instance from a pair of Cloud Connector instances. See [Install a Failover Instance for High Availability \[page 538\]](#).

The screenshot displays the 'Initial Setup' page of the SAP Cloud Connector Administration interface. At the top, there's a dark blue header with the SAP logo and 'Cloud Connector Administration'. On the right, a user dropdown shows 'Administrator'. Below the header, the page title 'Initial Setup' is on the left, and 'Save' and 'Help' icons are on the right. A blue message box states: 'You are required to change your password before being permitted to continue'. The main content area is divided into two sections: 'Mandatory Password Change' and 'Choose Installation Type'. The first section has three input fields: 'Current Password:', 'New Password:', and 'Repeat New Password:'. The second section has two radio buttons: 'Master (Primary Installation)' (selected) and 'Shadow (Backup Installation)', followed by a 'Description:' input field.

3. (Optional): When configuring a master, you can provide a (free-text) **Description** for this Cloud Connector instance that helps you distinguish different Cloud Connectors. This information will also be shown in the *Cloud Connectors* view in the SAP BTP cockpit.

User Administration

To edit the password for the **Administrator** user, choose **Configuration** from the main menu, tab **User Interface**, section **User Administration**:



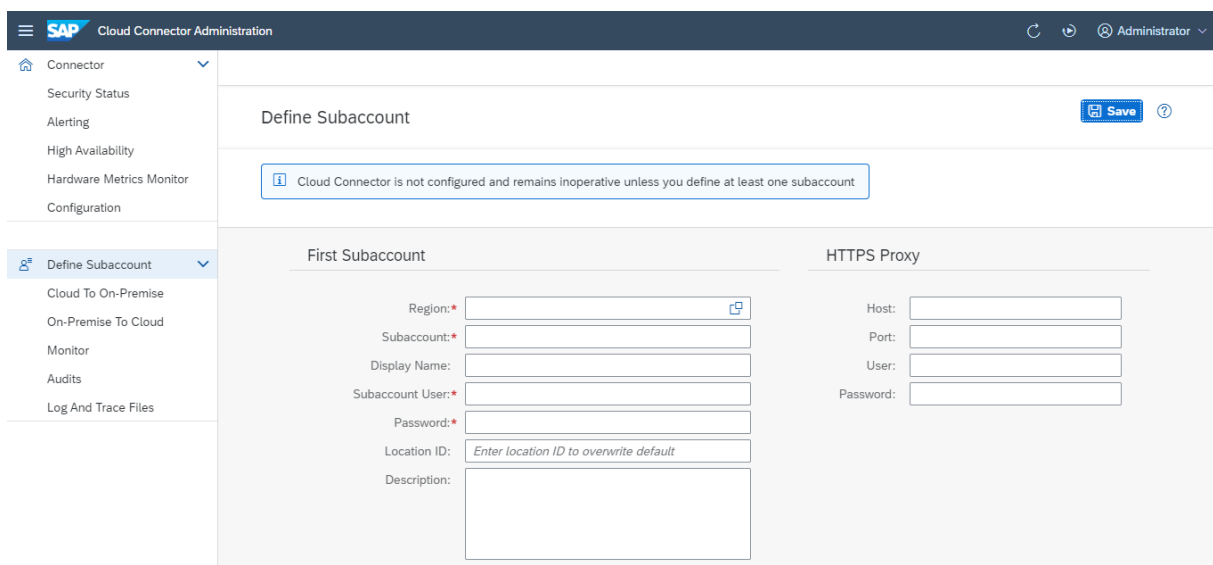
Note

User name and password cannot be changed at the same time. If you want to change the user name, you must enter only the current password in a first step. Do not enter values for **<New Password>** or **<Repeat New Password>** when changing the user name. To change the password in second step, enter the old password, the new one, and the repeated (new) password, but leave the user name unchanged.


Back to [Tasks \[page 278\]](#)

Set up Connection Parameters and HTTPS Proxy

When logging in for the first time, the following screen is displayed every time you choose an option from the main menu that requires a configured subaccount:



Note

If you want to skip the initial configuration, you can click the  icon in the upper right corner. You might need this in case of connectivity issues shown in your logs. You can add subaccounts later as described in [Managing Subaccounts \[page 291\]](#).

The Cloud Connector collects the following required information for your subaccount connection:

1. For **<Region>**, specify the SAP BTP region that should be used. You can choose it from the drop-down list, see [Regions](#).

Note

You can also configure a region yourself, if it is not part of the standard list. Either insert the region host manually, or create a custom region, as described in [Configure Custom Regions \[page 302\]](#).

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

Note

For the **Neo** environment, enter the subaccount's **technical name** in the field **<Subaccount>**, not the subaccount ID.

You can also add a new subaccount user with the role `Cloud Connector Admin` in the SAP BTP cockpit and use the new user and password.

Note

The Cloud Connector does not yet support *SAP Universal ID*. Please use your S-user or P-user credentials for the **<subaccount user>** and **<password>** fields instead.

For more information, see SAP note [3085908](#).

For the **Neo** environment, see [Add Members to Your Neo Subaccount](#).

For the **Cloud Foundry** environment, see [Add Org Members Using the Cockpit](#).

Tip

When using SAP Cloud Identity Services - Identity Authentication (IAS) as platform identity provider with two-factor authentication for your subaccount, you can simply append the required token to the regular password.

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 URI. To route requests to a Cloud Connector with a location ID, the location ID must be configured in the respective destinations.

Note

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

5. (Optional) Enter proxy host and port. Omit the proxy configuration unless your internal landscape is protected by a firewall that blocks any outgoing TCP traffic. In that case, you must specify an HTTPS proxy that the Cloud Connector can use to connect to SAP BTP.

The Cloud Connector performs two operations for which it may need a proxy in the situation outlined above:

- Downloading the correct connection configuration corresponding to your subaccount ID in SAP BTP.
- Establishing the TLS tunnel connection from the Cloud Connector to your SAP BTP subaccount.

Note

A proxy server is required to support TLS communication; a standard HTTP proxy will not suffice. Typically, you choose the same proxy settings as those being used by your standard Web browser.

Note

Some proxy servers require credentials for authentication. In this case, you need to provide the relevant user/password information.

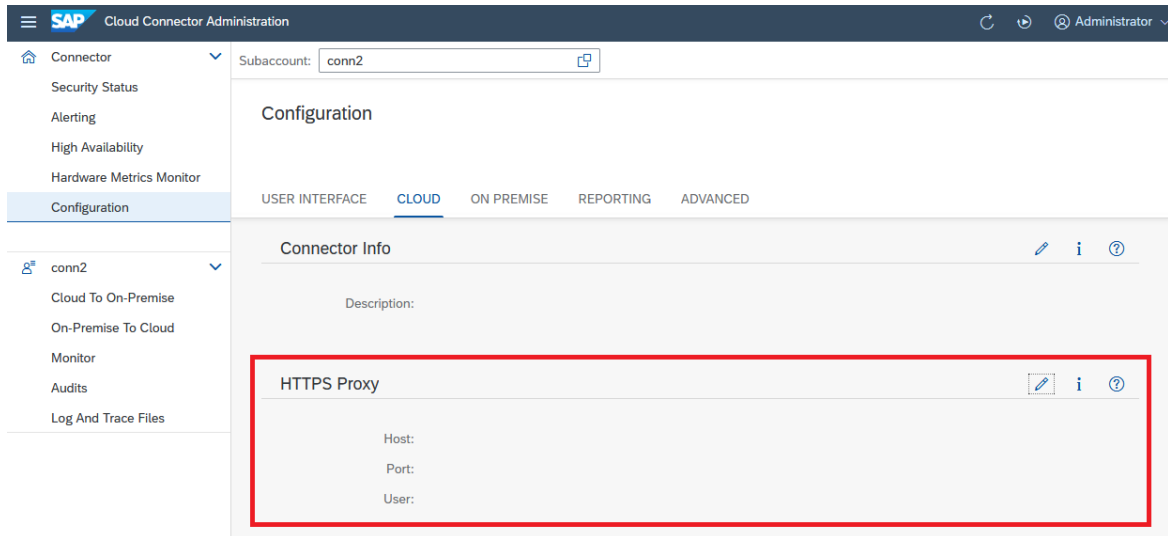
6. (Optional) You can provide a [Description](#) (free-text) of the subaccount that is shown when choosing the [Details](#) icon in the [Actions](#) column of the [Subaccount Dashboard](#). It lets you identify the particular Cloud Connector you use.
7. Choose [Save](#).

The Cloud Connector now starts a handshake with SAP BTP and attempts to establish a secure TLS tunnel to the server that hosts the subaccount in which your on-demand applications are running. However, no requests are yet allowed to pass from the cloud side to any of your internal backend systems. To allow your on-demand applications to access specific internal back-end systems, proceed with the access configuration described in the next section.

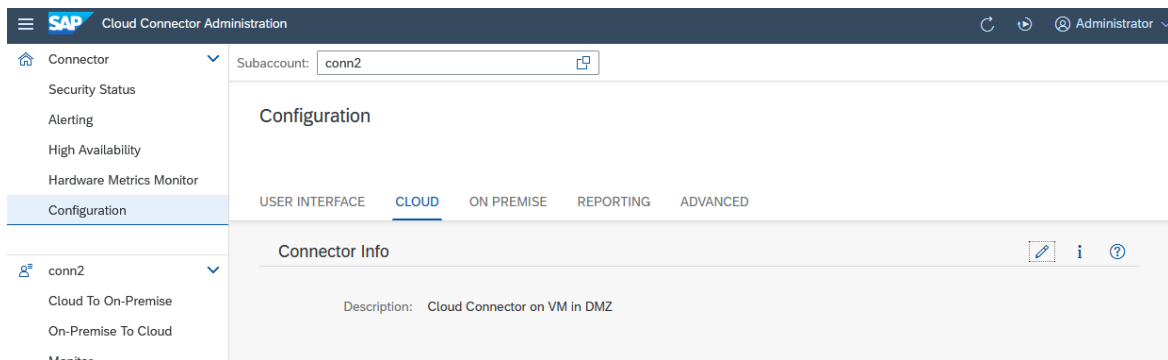
Note

The internal network must allow access to the port. Specific configuration for opening the respective port(s) depends on the firewall software used. The default ports are **80** for HTTP and **443** for HTTPS. For RFC communication, you must open a gateway port (default: **33+<instance number>**) and an arbitrary message server port. For a connection to a HANA Database (on SAP BTP) via JDBC, you must open an arbitrary *outbound* port in your network. Mail (SMTP) communication is not supported.

- If you later want to change your proxy settings (for example, because the company firewall rules have changed), choose [Configuration](#) from the main menu and go to the [Cloud](#) tab, section [HTTPS Proxy](#). Some proxy servers require credentials for authentication. In this case, you must provide the relevant user/password information.



- If you want to change the description for your Cloud Connector, choose [Configuration](#) from the main menu, go to the [Cloud](#) tab, section [Connector Info](#) and edit the description:

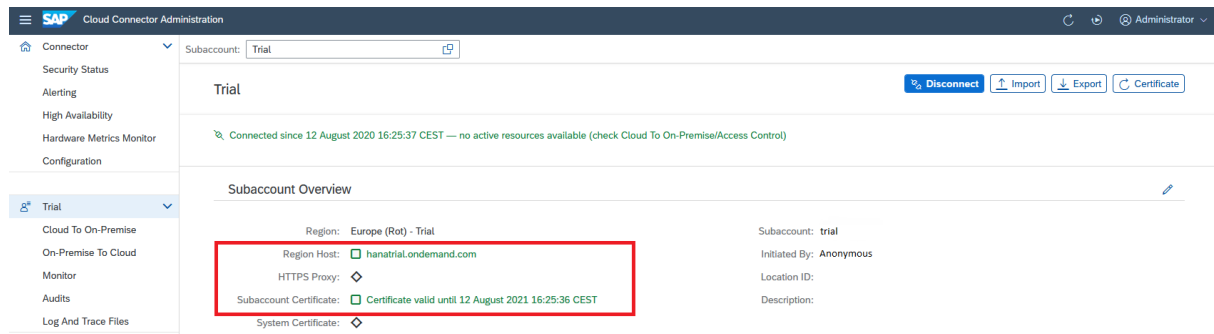


Back to [Tasks \[page 278\]](#)

Establish Connections to SAP BTP

As soon as the initial setup is complete, the tunnel to the cloud endpoint is open, but no requests are allowed to pass until you have performed the [Access Control](#) setup, see [Configure Access Control \[page 340\]](#).

To manually close (and reopen) the connection to SAP BTP, choose your subaccount from the main menu and select the [Disconnect](#) button (or the [Connect](#) button to reconnect to SAP BTP).



- The green icon next to *Region Host* indicates that it is valid and can be reached.
- If an *HTTPS Proxy* is configured, its availability is shown the same way. In the screenshot, the grey diamond icon next to *HTTPS Proxy* indicates that connectivity is possible without proxy configuration.

In case of a timeout or a connectivity issue, these icons are yellow (warning) or red (error), and a tooltip shows the cause of the problem. *Initiated By* refers to the user that has originally established the tunnel. During normal operations, this user is no longer needed. Instead, a certificate is used to open the connection to a subaccount.

- The status of the certificate is shown next to *Subaccount Certificate*. It is shown as valid (green icon), if the expiration date is still far in the future, and turns to yellow if expiration approaches according to your alert settings. It turns red as soon as it has expired. This is the latest point in time, when you should [Update the Certificate for Your Subaccount \[page 298\]](#).

Note

When connected, you can monitor the Cloud Connector also in the *Connectivity* section of the SAP BTP cockpit. There, you can track attributes like version, description and high availability set up. Every Cloud Connector configured for your subaccount automatically appears in the *Connectivity* section of the cockpit.

[Back to Tasks \[page 278\]](#)

Related Information

[Managing Subaccounts \[page 291\]](#)

[Initial Configuration \(HTTP\) \[page 285\]](#)

[Initial Configuration \(RFC\) \[page 287\]](#)

[Configuring the Cloud Connector for LDAP \[page 290\]](#)

[Managing Member Authorizations in the Neo Environment](#)

1.5.2.1.1 Initial Configuration (HTTP)

Configure the Cloud Connector for HTTP communication.

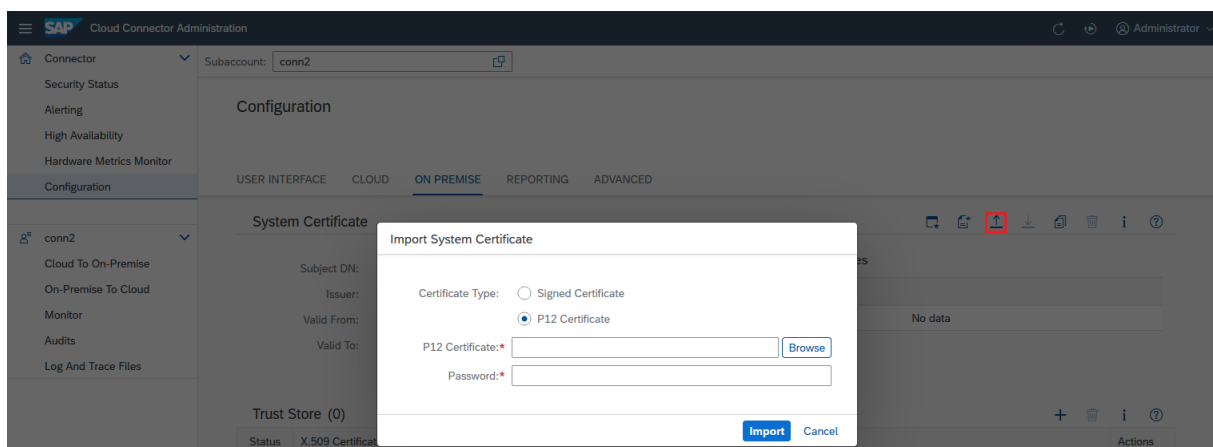
Installation of a System Certificate for Mutual Authentication

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 certificate authority (CA) that signed the Cloud Connector's system certificate must be trusted by all backend systems to which the Cloud Connector is supposed to connect.

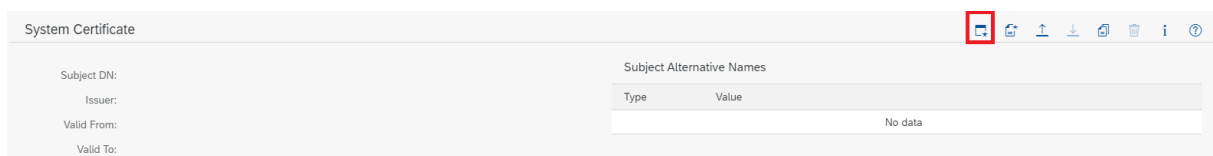
You must provide the system certificate as PKCS#12 file containing the client certificate, the corresponding private key and the CA root certificate that signed the client certificate (plus potentially the certificates of any intermediate CAs, if the certificate chain is longer than 2).

Procedure

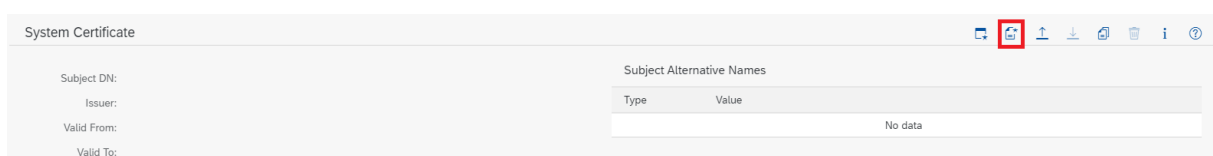
From the left panel, choose [Configuration](#). On the tab [On Premise](#), choose [System Certificate](#) [Import a certificate](#) to upload a certificate and provide its password:



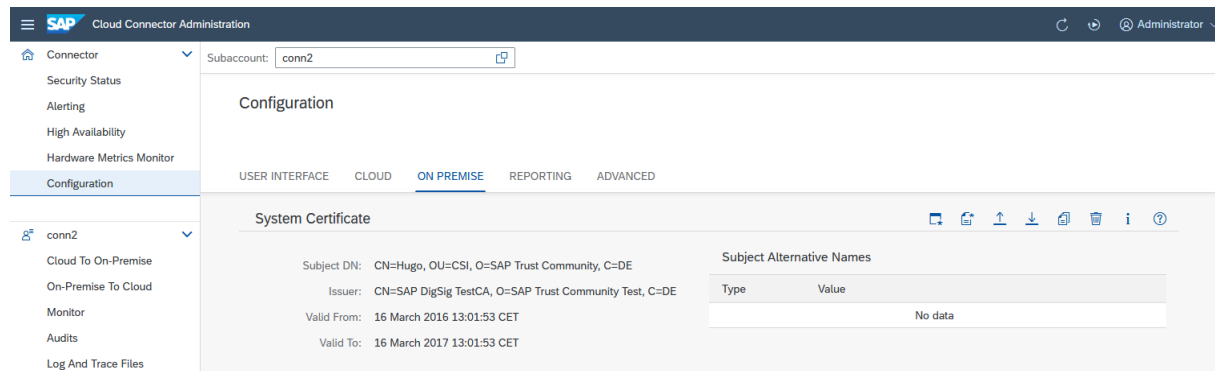
A second option is to start a *certificate signing request* procedure as described for the UI certificate in [Exchange UI Certificates in the Administration UI \[page 517\]](#) and upload the resulting signed certificate.



A third option is to generate a self-signed certificate. It might be useful if no CA is needed, for example, in a demo setup or if you want to use a dedicated CA. For this option, choose [Create and import a self-signed certificate](#):



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



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 \(HTTP\) \[page 342\]](#)

1.5.2.1.2 Initial Configuration (RFC)

Configure a Secure Network Connection (SNC) to set up the Cloud Connector for RFC communication to an ABAP backend system.

SNC Configuration for Mutual Authentication

To set up a mutual authentication between Cloud Connector and an ABAP backend system (connected via RFC), you can configure SNC for the Cloud Connector. It will then use the associated PSE for all RFC SNC requests. This means that the SNC identity, represented by this PSE, must:

- Be trusted by all backend systems to which the Cloud Connector is supposed to connect
- Play the role of a trusted external system by adding the SNC name of the Cloud Connector to the SNCSYSACL table. You can find more details in the SNC configuration documentation for the release of your ABAP system.

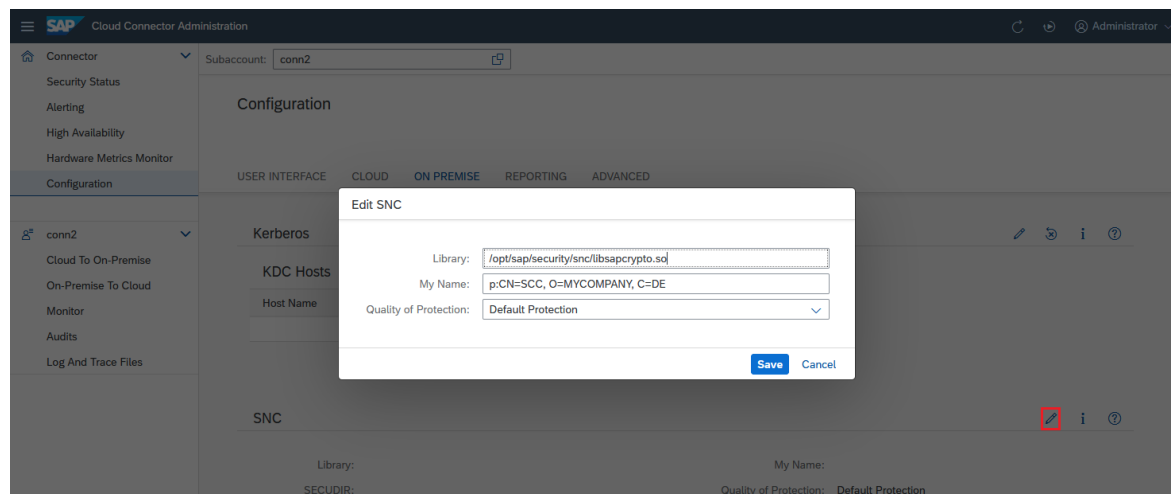
Prerequisites

- You have configured your ABAP system(s) for SNC. For detailed information on configuring SNC for an ABAP system, see also [Configuring SNC on AS ABAP](#).
- You have configured the ABAP System to trust the Cloud Connector's system SNC identity. To do this, or to establish trust for principal propagation, follow the steps described in [Configure Identity Propagation for RFC \[page 322\]](#).

Configuration Steps

1. Logon to the Cloud Connector
2. Choose [Configuration](#) from the main menu and go to tab [On Premise](#), section [SNC](#).
3. Enter the corresponding values in the fields [Library Name](#), [My Name](#), and [Quality of Protection](#)
4. Press [Save](#).

Example:



- [Library Name](#): Provides the location of the SNC library you are using for the Cloud Connector.

Note

Bear in mind that you must use one and the same security product on both sides of the communication.

- [My Name](#): The SNC name that identifies the Cloud Connector. It represents a valid scheme for the SNC implementation that is used.
- [Quality of Protection](#): Determines the level of protection that you require for the connectivity to the ABAP systems.

Note

When using **CommonCryptoLibrary** as SNC implementation, note [1525059](#) will help you to configure the PSE to be associated with the user running the Cloud Connector process.

Using the SAP Cryptographic Library

Note

This procedure is available as of Cloud Connector version 2.14.1

When using the SAP Cryptographic Library as SNC implementation, you can use the interactive setup scripts which can be found in the Cloud Connector's installation folder to ease the setup process.

Note

Using the scripts mentioned below is not mandatory for using the SAP cryptographic library. If you are familiar with the necessary steps needed for running a process with the desired identity, you can also configure the SNC setup for the SAP cryptographic library on your own.

Linux/Mac scripts: `snc_create_pse.sh`, `snc_import_ca_response.sh`

Windows scripts: `snc_create_pse.bat`, `snc_import_ca_response.bat`

1. Download and extract the SAP Cryptographic Library from the [Download Center](#) (search for `sapcryptolib`).
2. Copy the respective scripts depending on your OS to the SAP Cryptographic Library folder.
3. Make sure the Cloud Connector process is running.
4. Make sure the environment variable `SECUDIR` is properly set.
 1. For Linux, you can set it solely for the Cloud Connector process by extending the daemon as described in [Installation on Linux OS \[page 268\]](#).
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.pl0` has been created. This is the CSR you can use to get a signed certificate by your CA now. You can import the response certificate from your CA directly, or use the other script `import_ca_response` later. If you don't want to sign your certificate by a CA and use it as a self-signed certificate, choose `no`. Depending on your CA, you might need to provide, besides the signed certificate, all other intermediate certificates and the root certificate for the import. Copy them to the `SECUDIR` folder and specify them in the order shown below.
`Do you want to import a CA response now? [y/n] y`
`Specify the CA response file contained in D:\sec: ca_response.crt`

Specify further Root CAs (PEM, Base64 or DER binary) in D:\sec needed to complete the chain (separated by blanks), otherwise press enter: intermediate.crt root.crt
Importing CA response now...

...

CA-Response successfully imported into PSE "D:\sec\scc.pse"

10. Now, all other required steps are done, such as creating the credentials file `cred_v2`, importing the partner certificate and exporting the Cloud Connector's certificate to `SECUDIR` with the name `scc.crt` (which must be [imported](#) at the partner's side).

Create SSO server credentials...

...

Creation finished.

Export own certificate...

...


Export finished.

Import partner certificate into PSE file...

...

Completed.

11. Restart the Cloud Connector and check the above screen if `SECUDIR` is set correctly. SNC setup for SAP Cryptographic Library should be complete now. In a next step, you must [Configure Access Control \[page 350\]](#) and, if needed, [Configure Identity Propagation for RFC \[page 322\]](#).

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

Related Information

[Configure Identity Propagation for RFC \[page 322\]](#)

1.5.2.1.3 Configuring the Cloud Connector for LDAP

Configure the Cloud Connector to support LDAP in different scenarios (cloud applications using LDAP or Cloud Connector authentication).

Prerequisites

You have installed the Cloud Connector and done the basic configuration:

[Installation \[page 238\]](#)

[Initial Configuration \[page 278\]](#)

Steps

When using LDAP-based user management, you have to configure the Cloud Connector to support this feature. Depending on the scenario, you need to perform the following steps:

Scenario 1: Cloud applications using LDAP for authentication. Configure the destination of the LDAP server in the Cloud Connector: [Configure Access Control \(LDAP\) \[page 357\]](#).

Scenario 2: Internal Cloud Connector user management. Activate LDAP user management in the Cloud Connector: [Use LDAP for User Administration \[page 521\]](#).

1.5.2.2 Managing Subaccounts

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

Note

This topic refers to subaccount management in the Cloud Connector. If you are looking for information about managing subaccounts on SAP BTP (Cloud Foundry or Neo environment), see

- [Account Administration \(Cloud Foundry environment\)](#)
- [Administration and Operations, Neo Environment](#)

Context

You can connect to several subaccounts within a single Cloud Connector installation. Those subaccounts can use the Cloud Connector concurrently with different configurations. By selecting a subaccount from the drop-down box, all tab entries show the configuration, audit, and state, specific to this subaccount. In case of audit and traces, cross-subaccount info is merged with the subaccount-specific parts of the UI.

Note

We recommend that you group only subaccounts with the same qualities in a single installation:

- Productive subaccounts should reside on a Cloud Connector that is used for productive subaccounts only.
- Test and development subaccounts can be merged, depending on the group of users that are supposed to deal with those subaccounts. However, the preferred logical setup is to have separate development and test installations.

Prerequisites

You have assigned one of these roles/role collections to the subaccount user that you use for initial Cloud Connector setup, depending on the SAP BTP environment in which your subaccount is running:

Note

For the **Cloud Foundry** environment, you must know on which cloud management tools feature set (A or B) your account is running. For more information on feature sets, see [Cloud Management Tools — Feature Set Overview](#).

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

After establishing the Cloud Connector connection, this user is not needed any more, since it serves only for initial connection setup. You may revoke the corresponding role assignment then and remove the user from the [Members](#) list.

Note

If the Cloud Connector is installed in an environment that is operated by SAP, SAP provides a user that you can add as member in your SAP BTP subaccount and assign the required role.

Subaccount Dashboard

In the subaccount dashboard (choose your *Subaccount* from the main menu), you can check the state of all subaccount connections managed by this Cloud Connector at a glance.

Status	Subaccount	Display Name	Location ID	Region	Actions
✓	819986f5-460d-4a36-b...	qual	EMEA	Europe (Frankfurt) - AWS	⚙️ ✎️ 🗑️ ➡️
⚠️	ab1220dc-5bfb-45d1-9...	trial		Trial	⚙️ ✎️ 🗑️ ➡️
✓	ztglf9vgj9	Virtual Machine Test	EMEA	Trial	⚙️ ✎️ 🗑️ ➡️

In the screenshot above, the subaccount with display name *trial* (actual subaccount ID starts with *ab1220dc-5bfb-45d1-9...*) is already connected, but has no active resources exposed. All other subaccounts are connected with exposed resources and are fully operational. In addition, depending on the connection state, the dashboard allows you to do disconnect and connect subaccounts by pressing the respective button in the *Actions* column. You may also view details of, delete, or navigate to a subaccount using buttons from the *Actions* column.

The *Sort* buttons for the columns **Subaccount** and **Display Name** let you sort the entries by the column either ascending or descending, and the *Filter* buttons in the columns let you filter the listed entries.

You can use the *Filter* buttons above the dashboard to filter the shown subaccounts based on the connection status. You can select *all subaccounts*, *all connected ones*, *all disconnected ones*, *all subaccounts currently in connecting or reconnecting status*, or *all subaccounts for which establishing the connection has failed*.

If you want to connect an additional subaccount with your on-premise landscape, simply press the *Add Subaccount* button, which opens a dialog that is similar to the *Initial Configuration* [page 278] operation when establishing the first connection.

Add Subaccount

Region:*	<input type="text"/>
Subaccount:*	<input type="text"/>
Display Name:	<input type="text"/>
Subaccount User:*	<input type="text"/>
Password:*	<input type="password"/>
Location ID:	<input type="text" value="Enter location ID to overwrite default"/>
Description:	<input type="text"/>

Save

Cancel

Procedure

1. The `<Region>` field specifies the SAP BTP region that should be used, for example, **Europe (Rot)**. Choose the one you need from the drop-down list.

→ Remember

The available regions and region domains depend on the SAP BTP environment you are using. For more information, see [Regions](#) (Cloud Foundry and ABAP environment) or [Regions and Hosts Available for the Neo Environment](#).

→ Tip

You can also configure a region yourself, if it is not part of the standard list. Either insert the region host manually, or create a custom region, as described in [Configure Custom Regions \[page 302\]](#).

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

ⓘ Note

If your subaccount is on **Cloud Foundry**, you must enter the subaccount ID as `<Subaccount>`, rather than its actual (technical) name. For information on getting the subaccount ID, see [Find Your](#)

[Subaccount ID \(Cloud Foundry Environment\) \[page 301\]](#). As `<Subaccount User>` you must provide your **Login E-mail** instead of a user ID.

For the **Neo** environment, enter the subaccount's **technical name** in the field `<Subaccount>`, not the subaccount ID.

Alternatively, you can add a new subaccount user in the SAP BTP cockpit, assign the required authorization (see section **Prerequisites** above), and use the new user and password.

Note

The Cloud Connector does not yet support *SAP Universal ID*. Please use your S-user or P-user credentials for the `<subaccount user>` and `<password>` fields instead.

For more information, see SAP note [3085908](#).

→ Tip

When using SAP Cloud Identity Services - Identity Authentication (IAS) as platform identity provider with two-factor authentication (2FA / MFA) for your subaccount, you can simply append the required token to the regular password. For example, if your password is "eX7?6rUm" and the one-time passcode is "123456", you must enter "eX7?6rUm123456" into the `<Password>` field.

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:	<div></div>

Save

Cancel

- You can also delete a subaccount from the list of connections. The subaccount will be disconnected and all configurations will be removed from the installation.

Related Information

[Managing Member Authorizations in the Neo Environment](#)

[Copy a Subaccount Configuration \[page 296\]](#)

[Update the Certificate for a Subaccount \[page 298\]](#)

[Configure a Disaster Recovery Subaccount \[page 299\]](#)

[Find Your Subaccount ID \(Cloud Foundry Environment\) \[page 301\]](#)

[Configure Custom Regions \[page 302\]](#)

[Use a Custom IDP for Subaccount Configuration](#)

1.5.2.2.1 Copy a Subaccount Configuration

Copy an existing subaccount configuration in the Cloud Connector to another subaccount.

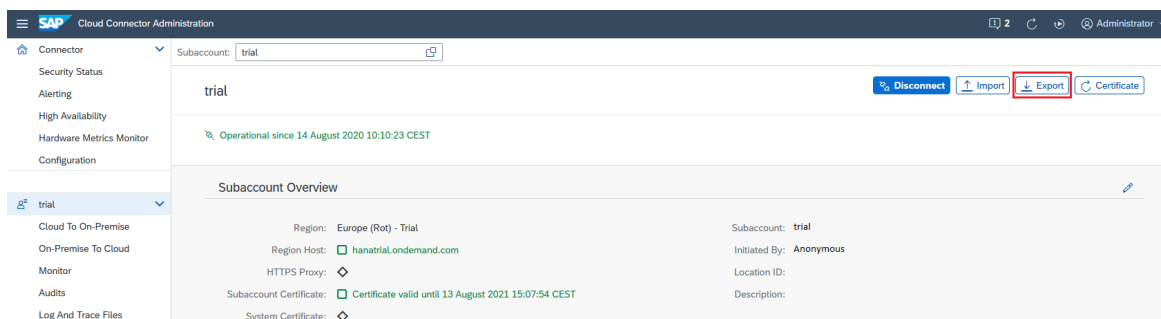
You can copy the configuration of a subaccount's *Cloud To On-Premise* and *On-Premise To Cloud* sections to a new subaccount, by using the export and import functions in the Cloud Connector administration UI.

Note

Principal propagation configuration (section *Cloud To On-Premise*) is not exported or imported, since it contains subaccount-specific data.

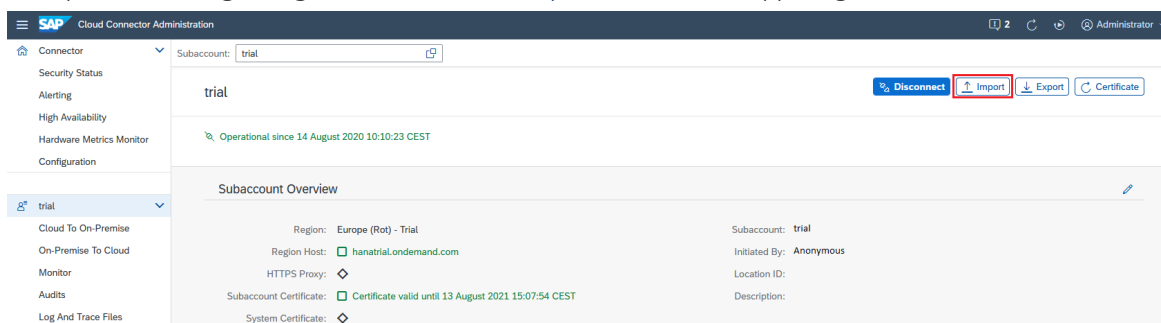
Procedure: Export an Existing Configuration

1. In the Cloud Connector administration UI, choose your subaccount from the navigation menu.
2. To export the existing configuration, choose the [Export](#) button in the upper right corner. The configuration is downloaded as a zip file to your local file system.



Procedure: Import an Existing Configuration

1. From the navigation menu, choose the subaccount to which you want to copy an existing configuration.
2. To import an existing configuration, choose the [Import](#) button in the upper right corner.



3. Select one of the following sources:
 1. [File](#), if you want to copy the configuration from a previously downloaded zip file.
 2. [Subaccount](#), if you want to copy the configuration directly from another existing subaccount.

Import Account Configuration

Source:

☒ File

☐ Subaccount

File:

Browse

Import

Cancel

1.5.2.2.2 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: User Roles \[page 9\]](#) (Neo environment)

Procedure

→ Tip

You can use this procedure even if the certificate has already expired.

Proceed as follows to update your subaccount certificate:

1. From the main menu, choose your subaccount.

📘 Note

To check the certificate's validity, click on the `<Subaccount Certificate>` in section [Subaccount Overview](#).

2. Choose the [Certificate](#) button. A dialog opens, requesting a user name and password.
3. Enter `<User Name>` and `<Password>` and choose [OK](#). The certificate assigned to your subaccount is refreshed.

📘 Note

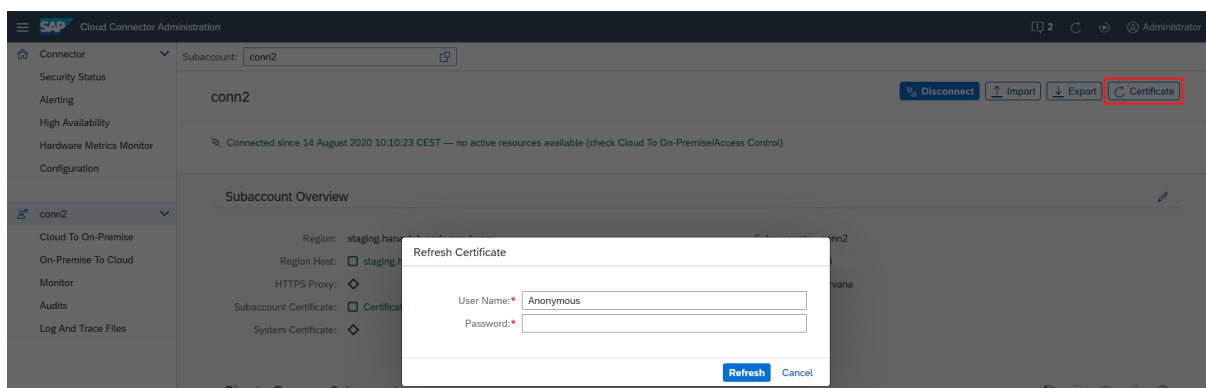
In the **Neo** environment, the user must have the role `Cloud Connector Admin Or Administrator`. For `<User Name>`, provide your [Login E-mail](#) or your user ID.

→ Tip

When using SAP Cloud Identity Services - Identity Authentication (IAS) as platform identity provider with two-factor authentication (2FA / MFA) for your subaccount, you can simply append the required

token to the regular password. For example, if your password is "eX7?6rUm" and the one-time passcode is "123456", you must enter "eX7?6rUm123456" into the <Password> field.

4. If you have configured a disaster recovery subaccount, go to section *Disaster Recovery Subaccount* below and choose *Refresh Disaster Recovery Certificate*.
5. Enter <User Name> and <Password> as in step 3 and choose *OK*.



1.5.2.2.3 Configure a Disaster Recovery Subaccount

Configure a subaccount as backup for disaster recovery.

⚠ Caution

This feature is deprecated.

Due to the discontinuation of the *Enhanced Disaster Recovery Service*, the related functionality in the Cloud Connector has been dropped as of version 2.16.

For more information, see [What's New for SAP Business Technology Platform](#).

Each subaccount (except trial accounts) can optionally have a disaster recovery subaccount.

Prerequisite is that you are using the enhanced disaster recovery.

The disaster recovery subaccount is intended to take over if the region host of its associated original subaccount faces severe issues.

A disaster recovery account inherits the configuration from its original subaccount except for the region host. The user can, but does not have to be the same.

Procedure

1. From the main menu, choose your subaccount.

2. In section *Disaster Recovery Subaccount*, choose *Configure disaster recovery subaccount*.
3. In the configuration dialog, select an appropriate *<Region Host>* from the drop-down list.

Note

The selected region host must be different from the region host of the original subaccount.

4. (Optional) You can adjust the *<Subaccount User>*.
5. Enter the *<Password>* for the subaccount user.
6. If configured, enter a *<Location ID>*.
7. Choose *Save*.

Note

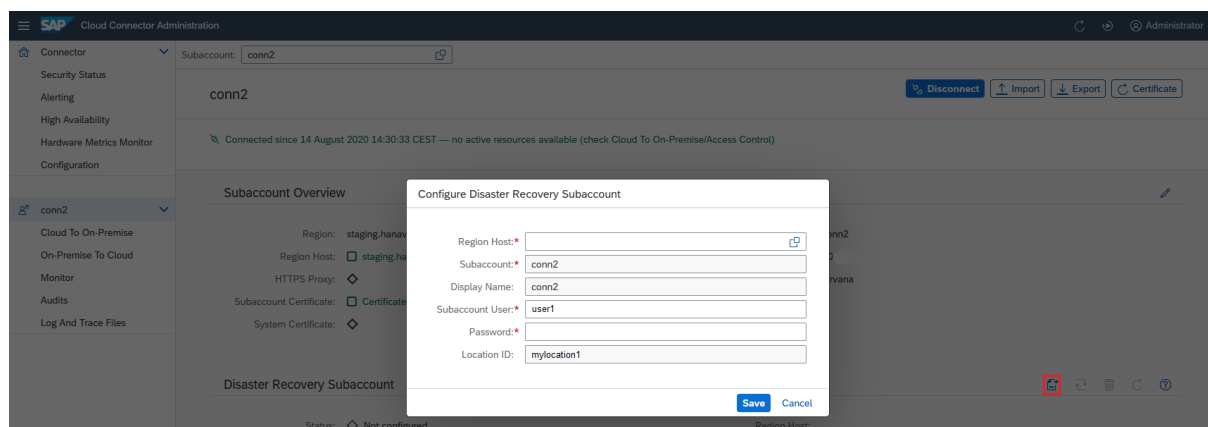
The technical subaccount name, the display name, and the location ID must remain the same. They are set automatically and cannot be changed.

Note

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

Note

If you want to change a disaster recovery subaccount, you must delete it first and then configure it again.



To switch from the original subaccount to the disaster recovery subaccount, choose *Employ disaster recovery subaccount*.

The disaster recovery subaccount then becomes active, and the original subaccount is deactivated.

You can switch back to the original subaccount as soon as it is available again.

Note

As of Cloud Connector 2.11, the cloud side informs about a disaster by issuing an event. In this case, the switch is performed automatically.

Related Information

[Convert a Disaster Recovery Subaccount into a Standard Subaccount \[page 301\]](#)

1.5.2.2.3.1 Convert a Disaster Recovery Subaccount into a Standard Subaccount

Convert a disaster recovery subaccount into a standard subaccount if the former primary subaccount's region cannot be recovered.

⚠ Caution

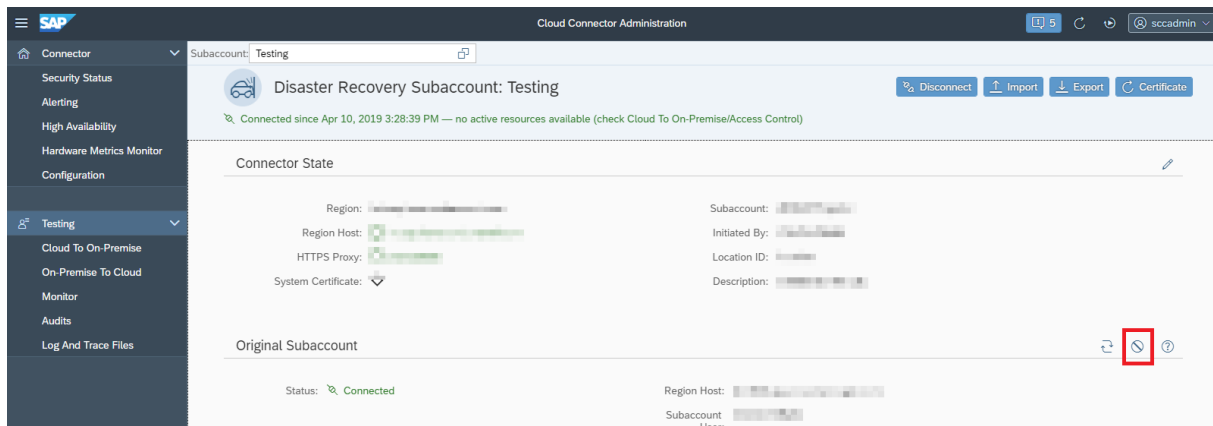
This feature is deprecated.

Due to the discontinuation of the *Enhanced Disaster Recovery Service*, the related functionality in the Cloud Connector has been dropped as of version 2.16.

For more information, see [What's New for SAP Business Technology Platform](#).

Disaster recovery subaccounts that were switched to disaster recovery mode can be elevated to standard subaccounts if a disaster recovery region replaces an original region that is not expected to recover.

If a disaster recovery subaccount should be used as primary subaccount, you can convert it by choosing the button *Discard original subaccount and replace it with disaster recovery subaccount*.

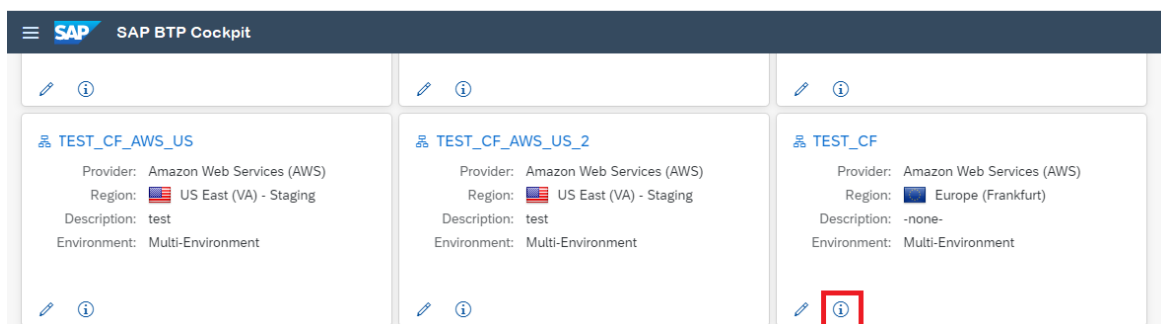


1.5.2.2.4 Find Your Subaccount ID (Cloud Foundry Environment)

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

In order to set up your subaccount in the Cloud Connector, you must know the subaccount ID. Follow these steps to acquire it:

1. Open the SAP BTP cockpit.
2. Navigate to the subaccount list of the global account containing your subaccount: choose ► [Home](#) ► [<Your Global Account>](#) ► [Account Explorer](#) ►.
3. Find your subaccount in the list.
4. Choose the [Info](#) icon in the subaccount tile to display the subaccount ID:



1.5.2.2.5 Configure Custom Regions

Configure regions that are not available in the standard selection.

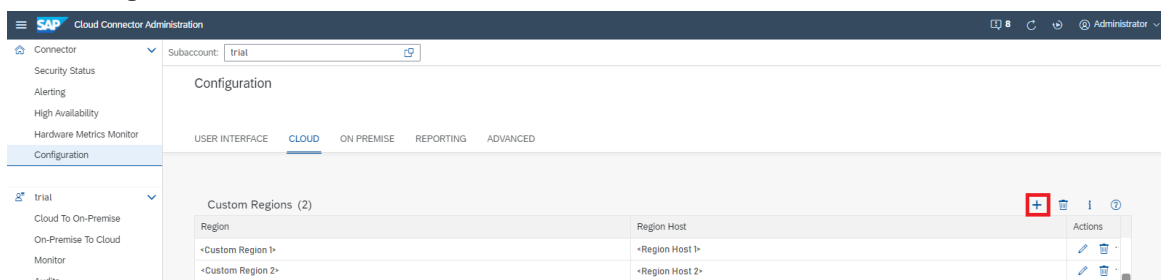
→ Tip

This procedure is useful in particular for regions introduced after the release of your current Cloud Connector version. Those regions are not included in the list of predefined regions.

If you want to use a custom region for your subaccount, you can configure regions in the Cloud Connector, which are not listed in the selection of standard regions.

To add a custom region, do the following:

1. From the Cloud Connector main menu, choose ► [Configuration](#) ► [Cloud](#) ► and go to the [Custom Regions](#) section.
2. To add a region to the list, choose the [Add](#) icon.



3. In the [Add Region](#) dialog, enter the [<Region>](#) and [<Region Host>](#) you want to use.
4. Choose [Save](#).

5. To edit a region from the list, select the corresponding line and choose the [Edit](#) icon.

1.5.2.3 Authenticating Users against On-Premise Systems

Authentication types supported by the Cloud Connector.

Currently, the Cloud Connector supports **basic authentication**, **principal propagation**, and **technical user propagation** as user authentication types towards internal systems. The destination configuration of the used cloud application defines which of these types is used for the actual communication to an on-premise system through the Cloud Connector, see [Managing Destinations](#).

- To use **basic authentication**, configure an on-premise system to accept basic authentication and to provide one or multiple service users. No additional steps are necessary in the Cloud Connector for this authentication type.
- To use **principal propagation** or **technical user propagation**, you must explicitly configure trust to those cloud entities from which user tokens are accepted as valid.
For more information, see [Configuring Principal Propagation \[page 304\]](#) and [Configuring Technical User Propagation \[page 339\]](#).
- When using HTTP as communication protocol, you can also perform a logon using the Cloud Connector system certificate, if there is no principal provided by the cloud side. The access control entry must be configured for this.

Note

For HTTP, also some other token-based authentication methods might work between the cloud application and the backend, depending on the backend capabilities.

Related Information

[Configuring Principal Propagation \[page 304\]](#)

[Configuring Technical User Propagation \[page 339\]](#)

1.5.2.3.1 Configuring Principal Propagation

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

Tasks in this section:

Task	Description
Set Up Trust [page 304]	Configure a trusted relationship in the Cloud Connector to support principal propagation. Principal propagation lets you forward the logged-on identity in the cloud to the internal system without requesting a password.
Configure a CA Certificate [page 308]	Install and configure an X.509 certificate to enable support for principal propagation.
Configuring Identity Propagation to an ABAP System [page 313]	Learn more about the different types of configuring and supporting principal propagation for a particular AS ABAP.
Configure Subject Patterns for Principal Propagation [page 327]	Define a pattern identifying the user for the subject of the generated short-lived X.509 certificate, as well as its validity period.
Configure a Secure Login Server [page 332]	Configuration steps for Java Secure Login Server (SLS) support.
Configure Kerberos [page 336]	The Cloud Connector lets you propagate users authenticated in SAP BTP via Kerberos against back-end systems. It uses the Service For User and Constrained Delegation protocol extension of Kerberos.
Configuring Identity Propagation to SAP NetWeaver AS for Java [page 338]	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 \[page 130\]](#) (Neo environment)

1.5.2.3.1.1 Set Up Trust

Establish trust to an identity provider to support principal propagation and technical user propagation.

Tasks

[Configure Trusted Entities in the Cloud Connector \[page 305\]](#)

[Configure an On-Premise System for Principal Propagation \[page 306\]](#)

[Trust Cloud Applications in the Cloud Connector \[page 307\]](#)

[Set up a Trust Store \[page 307\]](#)

Configure Trusted Entities in the Cloud Connector

Note

The information in this section applies to both principal propagation and technical user propagation.

You perform trust configuration to support principal propagation, that is, forwarding the *logged-on identity* in the cloud to the internal system without the need of providing the password. The same is done for technical user propagation, which logs on a *technical* user identified by an access token for an OAuth client without the need of providing the password.

By default, your Cloud Connector does not trust any entity that issues tokens for principal propagation. Therefore, the list of trusted identity providers (IdPs) is empty by default.

If you decide to use the principal propagation feature, you must establish trust to at least one IdP. The following IdP types are supported:

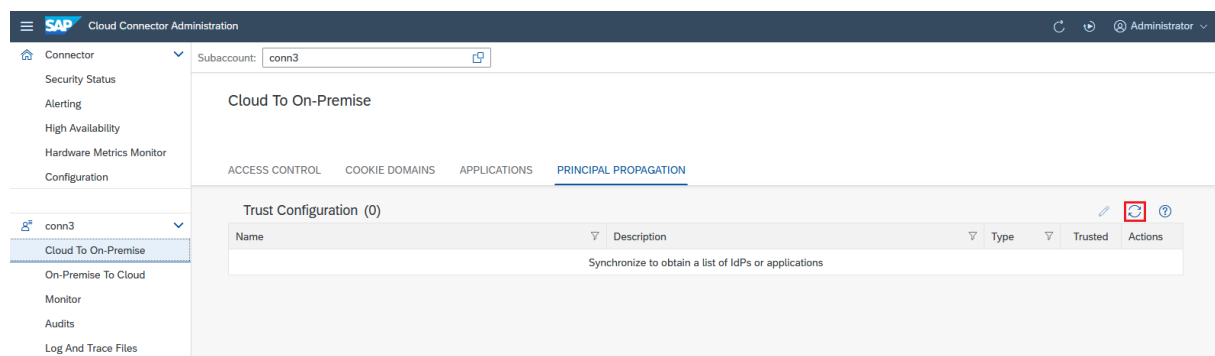
- **Neo environment:** *SAML* IdPs.

Note

In the Neo environment, you can also trust *HANA instances* and *Java applications* to act as IdPs.

- **Cloud Foundry environment:** *OAuth* IdPs.

You can configure trust to one or more IdPs per subaccount. After you've configured trust in the cockpit for your subaccount, for example, to your own company's IdP(s), you can synchronize this list with your Cloud Connector.



From your subaccount menu, choose [Cloud to On-Premise](#) and go to the [Principal Propagation](#) tab. Choose the [Synchronize](#) button to store the list of existing identity providers locally in your Cloud Connector.

Select an entry to see its details:

- **Name:** the name associated with the identity provider.
- **Description:** descriptive information about this entry.
- **Type:** type of the trusted entity.
- **Trusted:** indicates whether the entry is trusted for principal propagation.
- **Actions:** Choose the [Show Certificate Information](#) icon to display detail information for the corresponding entry. The Cloud Connector runtime will use the certificate associated with the entry to verify that the assertion used for principal propagation was issued by a trusted entity.

You can decide for each entry, whether to trust it for the principal propagation use case by choosing [Edit](#) and (de)selecting the [Trusted](#) checkbox.

Note

Whenever you update a SAML IdP configuration for a subaccount on cloud side, you must synchronize the trusted entities in the Cloud Connector. Otherwise the validation of the forwarded SAML assertion will fail with an exception containing an exception message similar to this:
Caused by: com.sap.engine.lib.xml.signature.SignatureException: Unable to validate signature -> java.security.SignatureException: Signature decryption error: javax.crypto.BadPaddingException: Invalid PKCS#1 padding: encrypted message and modulus lengths do not match!.

For more information, see also [Include Tokens from Corporate Identity Providers or Identity Authentication in Tokens of the SAP Authorization and Trust Management Service](#).

Back to [Tasks \[page 305\]](#)

Configure an On-Premise System for Principal Propagation

Set up principal propagation from SAP BTP to your internal system that is used in a hybrid scenario.

Note

As a prerequisite for principal propagation for RFC, the following cloud application runtime versions are required:

- for Java Web: **1.51.8** or higher
- for Java EE 6 Web Profile: **2.31.11** or higher
- other runtimes support it with any version

1. Set up trust to an entity that is issuing an assertion for the logged-on user (see section above).
2. Set up the system identity for the Cloud Connector.
 - For HTTPS, you must import a system certificate into your Cloud Connector.
 - For RFC, you must import an SNC PSE into your Cloud Connector.

3. Configure the target system to trust the Cloud Connector.

There are two levels of trust:

1. First, you must allow the Cloud Connector to identify itself with its system certificate (for HTTPS), or with the SNC PSE (for RFC).
2. Then, you must allow this identity to propagate the user accordingly:
 - For HTTPS, the Cloud Connector forwards the true identity in a short-lived x.509 certificate in an HTTP header named **SSL_CLIENT_CERT**. The system must use this certificate for logging on the real user. The SSL handshake, however, is performed through the system certificate. For more information on identity forwarding, see [Configure Access Control \(HTTP\) \[page 342\]](#).
 - For RFC, the Cloud Connector forwards the true identity as part of the RFC protocol.

For more information, see [Configuring Identity Propagation to an ABAP System \[page 313\]](#).

4. Configure the user mapping in the target system. The x.509 certificate contains information about the cloud user in its subject. Use this information to map the identity to the appropriate user in this system. This step applies for both HTTPS and RFC.

Note

If you have the following scenario: *Application1->AppToAppSSO->Application2->Principal Propagation->On premise Backend System* you must mark *Application2* as **trusted** by the Cloud Connector in tab *Principal Propagation*, section *Trust Configuration*.

If you use an identity provider that issues unsigned assertions, you must mark all relevant applications as **trusted** by the Cloud Connector in tab *Principal Propagation*, section *Trust Configuration*.

Back to [Tasks \[page 305\]](#)

Trust Cloud Applications in the Cloud Connector

Configure an allowlist for trusted cloud applications, see [Configure Trust \[page 501\]](#).

Back to [Tasks \[page 305\]](#)

Set up a Trust Store

Configure a trust store that acts as an allowlist for trusted on-premise systems. See [Configure Trust \[page 501\]](#).

Back to [Tasks \[page 305\]](#)

Related Information

[Principal Propagation](#) (Cloud Foundry)

[Principal Propagation \[page 130\]](#) (Neo)

1.5.2.3.1.2 Configure a CA Certificate

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

Note

The information in this section applies to both principal propagation and technical user propagation.

Supported CA Mechanisms

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

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

Note

Prior to version 2.7.0, this was the only option and the system certificate was acting both as client certificate and CA certificate in the context of principal propagation.

- Using a *Secure Login Server* (SLS) and delegate the CA functionality to it.

The Cloud Connector uses the configured CA approach to issue short-lived certificates for logging on the same identity in the back end that is logged on in the cloud. For establishing trust with the back end, the respective configuration steps are independent of the approach that you choose for the CA.

Install a local CA Certificate

To issue short-lived certificates that are used for principal propagation to a back-end system, you can import an X.509 client certificate into the Cloud Connector. This CA certificate must be provided as *PKCS#12* file containing the (intermediate) certificate, the corresponding private key, and the CA root certificate that signed the intermediate certificate (plus the certificates of any other intermediate CAs, if the certificate chain includes more than those two certificates).

Use either of the following options to install a local CA certificate:

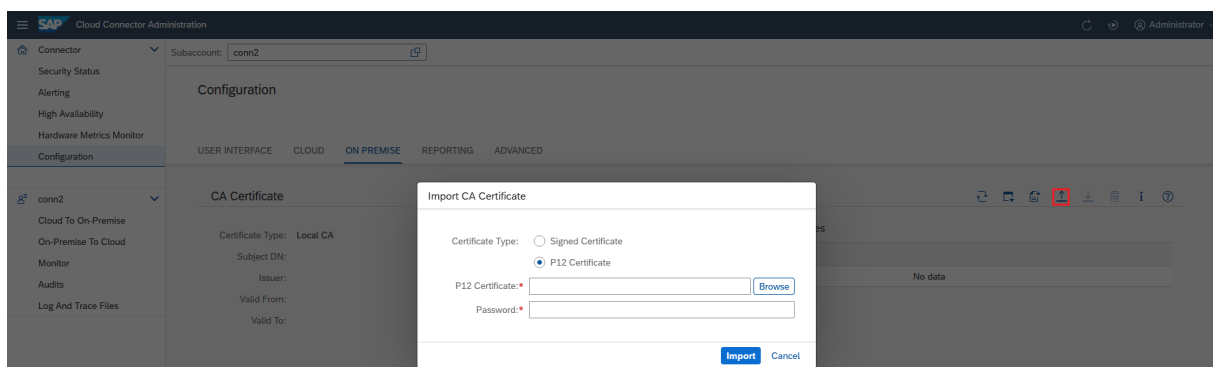
- Option 1: Choose the PKCS#12 file from the file system, using the file upload dialog. For the import process, you must also provide the file password.

- Option 2: Start a *Certificate Signing Request (CSR)* procedure like for the UI certificate, see [Exchange UI Certificates in the Administration UI \[page 517\]](#).
- 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*.

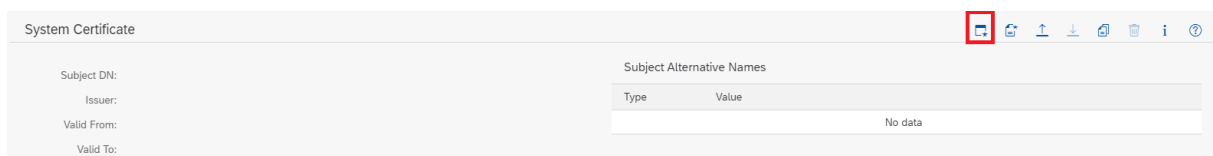
⚠ Caution

The CA certificate should have the `keyUsage` attribute `keyCertSign`. Many systems verify that the issuer of a certificate has this attribute and deny a client certificate, if this attribute is not present. When using the *Certificate Signing Request* procedure, the attribute will be requested for the CA certificate. Also, when generating a self-signed certificate, this attribute will be added automatically.

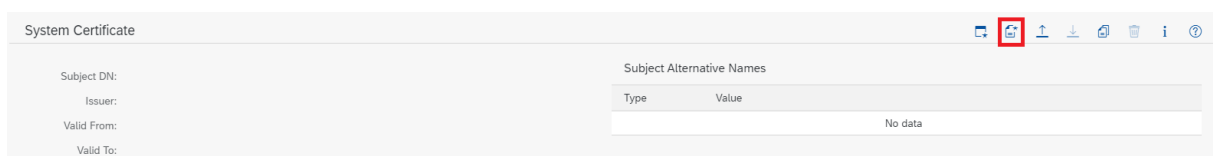
Choose *Import a certificate* for option 1:



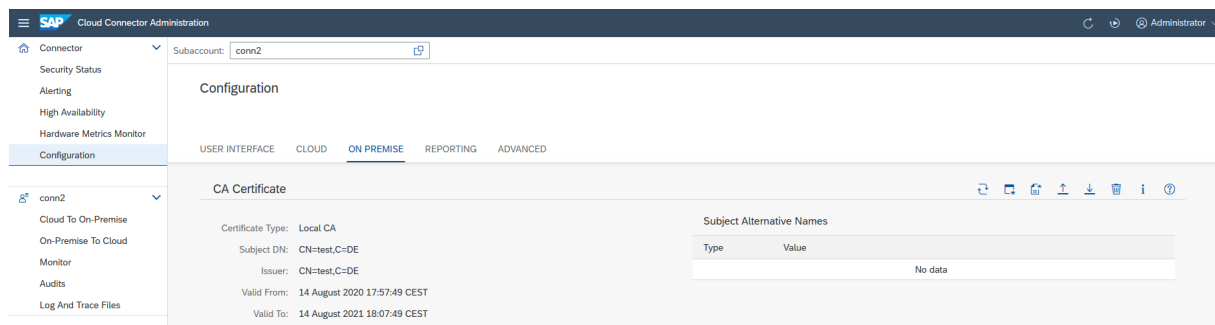
Choose *Generate a certificate signing request* for option 2:



Choose *Create and import a self-signed certificate* if you want to use option 3:



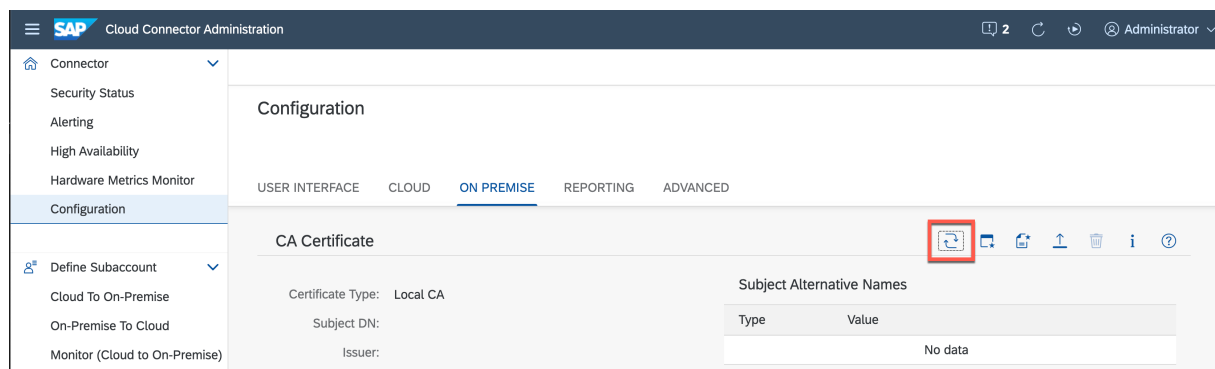
In particular for this option, it is useful to export the public key of the CA by choosing the respective button. After successful import of the CA certificate, its distinguished name, the name of the issuer, and the validity dates are shown:



If a CA certificate is no longer required, you can delete it. Use the respective [Delete](#) button and confirm the deletion.

Configure a CA Hosted by a Secure Login Server

If you want to delegate the CA functionality to a *Secure Login Server* (SLS), choose the CA using the [Secure Login Server](#) option and configure the SLS as follows, after having configured the SLS as described in [Configure a Secure Login Server \[page 332\]](#).



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

Configure CA Via Secure Login Server

i You may leave this field empty and go to the next step to set up the configuration without metadata URL

Meta URL:

Previous **Next** Cancel

Using the metadata URL lets you fetch the most relevant data from SLS instance. You only have to choose the profile configured on SLS, which should be used for the generation of short-lived certificates. Choose *Finish* to save the configuration.

Configure CA Via Secure Login Server

Profile ID: *

.....

.....

.....

Previous **Finish** Cancel

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.

Note

Used if SLS API *version 3* is configured.

- **Profile:** SLS profile that allows to issue certificates as needed for principal propagation with the Cloud Connector.

Note

Used if SLS API *version 2* is configured.

- **Authentication Port:** Port over which the Cloud Connector is requesting the short-lived certificates from SLS.

Note

For this privileged port, a client certificate authentication is required, for which the Cloud Connector's system certificate is used.

Configure CA Via Secure Login Server

API version: *

3

Host Name: *

.....

Port:

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: *

.....

Authentication Port:

10443

Previous

Finish

Cancel

Choose *Finish* to save the configuration.

Related Information

[Configure a Secure Login Server \[page 332\]](#)

[Initial Configuration \(HTTP\) \[page 285\]](#)

[Initial Configuration \(RFC\) \[page 287\]](#)

1.5.2.3.1.3 Configuring Identity Propagation to an ABAP System

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

Note

The information in this section applies to both principal propagation and technical user propagation.

Task	Description
Configure Identity Propagation for HTTPS [page 314]	Step-by-step instructions to configure principal propagation to an ABAP server for HTTPS.
Configure Identity Propagation via SAP Web Dispatcher [page 319]	Set up a trust chain to use principal propagation to an ABAP server for HTTPS via SAP Web Dispatcher.

Task	Description
Configure Identity Propagation for RFC [page 322]	Step-by-step instructions to configure principal propagation to an ABAP server for RFC.
Rule-Based Mapping of Certificates [page 325]	Map short-lived certificates to users in the ABAP server.

1.5.2.3.1.3.1 Configure Identity Propagation for HTTPS

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

Note

The information in this section applies to both principal propagation and technical user propagation. These two types of user propagation are combined as *identity* propagation.

Example Data

The following data are used in this example:

- System certificate was issued by: **CN=MyCompany CA,O=Trust Community,C=DE**.
- It has the subject: **CN=SCC,OU=BTP Scenarios,O=Trust Community,C=DE**.
- The short-lived certificate has the subject **CN=P1234567890**, where P1234567890 is the platform user.

Tasks

[Prerequisites \[page 314\]](#)

[Configure an ABAP System to Trust the Cloud Connector's System Certificate \[page 315\]](#)

[Map Short-Lived Certificates to Users \[page 317\]](#)

[Access ICF Services \[page 317\]](#)

Prerequisites

- For identity propagation, mutual authentication (mTLS) is mandatory.
To enable mTLS, you must configure a system certificate as described in this topic. Follow step 1 below to make sure the ABAP system trusts this certificate. You can use the connection check and the [Details](#) button to check if mTLS is working.

- The access control entry (see [Configure Access Control \(HTTP\) \[page 342\]](#)) must have specified the respective identity type to forward the identity correctly.

To perform the following steps, you must have the corresponding authorizations in the ABAP system for the transactions mentioned below (administrator role according to your specific authorization management) as well as an administrator user for the Cloud Connector.

Back to [Tasks \[page 314\]](#)

Back to [Example Data \[page 314\]](#)

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

This step includes two sub-steps:

[Configure the ABAP system to trust the Cloud Connector's system certificate \[page 315\]](#)

[Configure the Internet Communication Manager \(ICM\) to trust the system certificate for principal propagation \[page 315\]](#)

Configure the ABAP system to trust the Cloud Connector's system certificate:

1. Open the *Trust Manager* (transaction STRUST).
2. Double-click the *SSL-Server Standard* folder in the menu tree on the left.
3. In the displayed screen, click the *Import certificate* button.
4. In the dialog window, choose the certificate file representing the public key of the issuer of the system certificate, for example, in DER format. Typically, this is a CA certificate. If you decide to use a self-signed system certificate, it is the system certificate itself.
5. The details of this certificate are shown in the section above. Using the example certificate data, you would see **CN=MyCompany CA, O=Trust Community, C=DE** as subject.
6. If you are sure that you are importing the correct certificate, you can integrate the certificate into the certificate list by choosing the *Add to Certificate List* button.
7. Now, the CA certificate (**CN=MyCompany CA, O=Trust Community, C=DE**) is part of the certificate list.

Back to [Step \[page 315\]](#)

Configure the Internet Communication Manager (ICM) to trust the system certificate for identity propagation:

1. Open the *Profile Editor* (transaction RZ10).
2. Select the profile you want to edit, for example, the DEFAULT profile.
3. Select the *Extended maintenance* radio button and choose the *Change* button.
4. Create the following parameter: `icm/trusted_reverse_proxy_<x> = SUBJECT="<subject>" , ISSUER="<issuer>"`.
 - Select a free index for **<x>**.
 - **<subject>** is the subject of the system certificate (example data: **CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE**).

- **<issuer>** is the issuer of the system certificate (example data: **CN=MyCompany CA, O=Trust Community, C=DE**).
- Example: `icm/trusted_reverse_proxy_2 = SUBJECT="CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE", ISSUER="CN=MyCompany CA, O=Trust Community, C=DE".`

Note

If your ABAP system uses kernel 7.42 or lower, see SAP note [2052899](#) or set the following two parameters:

- `icm/HTTPS/trust_client_with_issuer`: this is the issuer of the system certificate (example data: **CN=MyCompany CA, O=Trust Community, C=DE**).
- `icm/HTTPS/trust_client_with_subject`: this is the subject of the system certificate (example data: **CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE**).

Caution

The ICM expects *blanks after the separating comma* for the issuer and subject elements. So, even if the Cloud Connector administration UI shows a string without blanks, for example: `CN=SCC,OU=BTP Scenarios,O=Trust Community,C=DE`, you must specify in ICM: `CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE`.

Caution

Make sure that `icm/HTTPS/verify_client` is set to either **1** (request certificate) or **2** (require certificate). If the parameter is set to **0**, trust cannot be established.

5. Save the profile.
6. Open the *ICM Monitor* (transaction `SMICM`) and restart the ICM by choosing **Administration > ICM > Exit Hard > Global**.
7. Verify that the two profile parameters have been taken over by ICM by choosing **Goto > Parameters > Display**.

Note

If you have an SAP Web Dispatcher installed in front of the ABAP system, trust must be added in its configuration files with the same parameters as for the ICM. Also, you must add the system certificate of the Cloud Connector to the trust list of the Web dispatcher Server PSE. For more information, see [Configure Identity Propagation via SAP Web Dispatcher \[page 319\]](#).

Caution

When using identity propagation with X.509 certificates, you cannot use the strict mode in certificate block management (transaction code: `CRCONFIG`) for the CRL checks within profile `SSL_SERVER`.

Back to [Step \[page 315\]](#)

Back to [Tasks \[page 314\]](#)

Back to [Example Data \[page 314\]](#)

2. Map Short-Lived Certificates to Users

For systems later than SAP NetWeaver 7.3 EHP1 (7.31), you can use rule-based certificate mapping, which is the recommended way to create the required user mappings. For more information, see [Rule-Based Mapping of Certificates \[page 325\]](#).

In older releases (for which this feature does not exist yet), you can do this manually in the system as described below, or use an identity management solution generating the mapping table for a more comfortable approach.

1. Open [Assignment of External ID to Users](#) (transaction `EXTID_DN`).
2. Switch to the edit mode.
3. Create a new entry. Specify the subject of the certificate as `External ID`. When using the example data, this is `CN=P1234567890`. In the `User` field, provide the appropriate ABAP user, for example `JOHNSMITH`.
4. Choose [Activate](#).
5. Save the mapping.
6. Repeat the previous steps for all users that should be supported for the scenario.

Back to [Tasks \[page 314\]](#)

Back to [Example Data \[page 314\]](#)

3. Access ICF Services

To access the required ICF services for your scenario in the ABAP system, choose one of the following procedures:

For Cloud Connector version 2.15 or higher:

- To access ICF services via **certificate logon** (using the system certificate), select [Allow Principal Propagation](#), choose the principal type **x.509 Certificate**, and select [System Certificate for Logon](#) in the corresponding system mapping.
This setting lets you use the system certificate for trust in case of principal propagation as well as for user authentication. For details, see [Configure Access Control \(HTTP\) \[page 342\]](#), step 7, 8 (*Procedure for Cloud Connector version 2.15 and higher*), and 9.
Additionally, make sure that all required ICF services allow *Logon Through SSL Certificate* as logon method.
- To access ICF services via the logon method **Basic Authentication** (logon with user/password) and identity propagation, select [Allow Principal Propagation](#), choose the principal type **x.509 Certificate**, and do *not* select [System Certificate for Logon](#) in the corresponding system mapping.
This setting lets you use the system certificate for trust, but prevents its usage for user authentication. For details, see [Configure Access Control \(HTTP\) \[page 342\]](#), step 7, 8 (*Procedure for Cloud Connector version 2.15 and higher*), and 9.
Additionally, make sure that all required ICF services allow *Basic Authentication* and *Logon Through SSL Certificate* as logon method.
- If some of the ICF services require `Basic Authentication`, while others should be accessed via system certificate logon, perform these steps:

1. In the Cloud Connector system mapping, select [Allow Principal Propagation](#), choose the principal type **X.509 Certificate**, and select [System Certificate for Logon](#) in the corresponding system mapping as described above.
2. In the ABAP system, choose transaction code SICF and go to [Maintain Services](#).
3. Select the service that requires Basic Authentication as logon method.
4. Double-click the service and go to tab [Logon Data](#).
5. Switch to [Alternative Logon Procedure](#) and ensure that the Basic Authentication logon procedure is listed before Logon Through SSL Certificate.

For Cloud Connector versions below 2.15:

- To access ICF services via certificate logon, choose the principal type **X.509 Certificate (general usage)** in the corresponding system mapping. This setting lets you use the system certificate for trust as well as for user authentication.
For details, see [Configure Access Control \(HTTP\) \[page 342\]](#), step 8 (*Procedure for Cloud Connector versions below 2.15*).
Additionally, make sure that all required ICF services allow Logon Through SSL Certificate as logon method.
- To access ICF services via the logon method Basic Authentication (logon with user/password) and principal propagation, choose the principal type **X.509 Certificate (strict usage)** in the corresponding system mapping. This setting lets you use the system certificate for trust, but prevents its usage for user authentication.
For details, see [Configure Access Control \(HTTP\) \[page 342\]](#), step 8 (*Procedure for Cloud Connector versions below 2.15*).
Additionally, make sure that all required ICF services allow Basic Authentication and Logon Through SSL Certificate as logon methods.
- If some of the ICF services require Basic Authentication, while others should be accessed via system certificate logon, perform these steps:
 1. In the Cloud Connector system mapping, choose the principal type **X.509 Certificate (general usage)** as described above.
 2. In the ABAP system, choose transaction code SICF and go to [Maintain Services](#).
 3. Select the service that requires Basic Authentication as logon method.
 4. Double-click the service and go to tab [Logon Data](#).
 5. Switch to [Alternative Logon Procedure](#) and ensure that the Basic Authentication logon procedure is listed before Logon Through SSL Certificate.

Note

If you are using SAP Web Dispatcher for communication, you must configure it to forward the SSL certificate to the ABAP backend system, see [Forward SSL Certificates for X.509 Authentication](#) (SAP Web Dispatcher documentation).

Back to [Tasks \[page 314\]](#)

Back to [Example Data \[page 314\]](#)

Related Information

[Configure Identity Propagation via SAP Web Dispatcher \[page 319\]](#)

[Rule-Based Mapping of Certificates \[page 325\]](#)

[Configure Subject Patterns for Principal Propagation \[page 327\]](#)

[Set Up Trust \[page 304\]](#)

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

[Principal Propagation \[page 130\]](#) (Neo environment)

1.5.2.3.1.3.1.1 Configure Identity Propagation via SAP Web Dispatcher

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

Note

The information in this section applies to both principal propagation and technical user propagation. These two types of user propagation are combined as *identity* propagation.

Concept

If you are using an intermediate SAP Web Dispatcher to connect to your ABAP backend system, you must set up a trust chain between the involved components Cloud Connector, SAP Web Dispatcher, and ABAP backend system.

Before configuring the ABAP system (see [Configure Identity Propagation for HTTPS \[page 314\]](#)), in a first step you must configure SAP Web Dispatcher to accept and forward user principals propagated from a cloud account to an ABAP backend.

To do this, follow the step-by-step instructions below.

Example Data

The following data and setup is used for this scenario:

- System certificate was issued by CN=MyCompany CA, O=Trust Community, C=DE
- It has the subject CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE

Tasks

- [Prerequisites \[page 320\]](#)
- [Configure the SAP Web Dispatcher to Trust the Cloud Connector's System Certificate \[page 320\]](#)
- [Next Steps \[page 322\]](#)

Prerequisites

- Your SAP Web Dispatcher version is 7.53 or higher. See SAP note [908097](#) for information on recommended SAP Web Dispatcher versions.
- We recommend that you use a standalone Web Dispatcher deployment. To learn about deployment options, see SAP note [3115889](#).
- Make sure your SAP Web Dispatcher supports SSL. See [Configure SAP Web Dispatcher to Support SSL](#).
- Ensure that TLS client certificates can be used for authentication in the backend system. See [How to Configure SAP Web Dispatcher to Forward SSL Certificates for X.509 Authentication](#) for step-by-step instructions.

Back to [Tasks \[page 320\]](#)

Back to [Concept \[page 319\]](#)

Configure SAP Web Dispatcher to Trust the Cloud Connector's System Certificate

To allow Cloud Connector client certificates for authentication in the backend system, perform the following two steps:

1. **Configure SAP Web Dispatcher to trust the Cloud Connector's system certificate:**
 1. To import the system certificate to SAP Web Dispatcher, open the SAP Web Dispatcher administration interface in your browser.

Note

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

2. In the menu, navigate to [SSL and Trust Configuration](#) and select [PSE Management](#).
3. In the **Manage PSE** section, select [SAPSSLS.pse](#) from the drop-down list. By default, SAPSSLS.pse contains the server certificate and the list of trusted clients that SAP Web Dispatcher trusts as a server.
4. In the **Trusted Certificates** section, choose [Import Certificate](#).
5. Enter the certificate as *base64-encoded* into the text box. The procedure to export your certificate in such a format is described in [Forward SSL Certificates for X.509 Authentication](#), step 1.

Note

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

6. Choose *Import*.

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

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

- Create or edit the following parameter in SAP Web Dispatcher:
`icm/trusted_reverse_proxy_<x> = SUBJECT="<subject>", ISSUER="<issuer>"`
 - Select a free index for <x>.
 - <subject>: Subject of the system certificate (example data: CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE)
 - <issuer>: Issuer of the system certificate (example data: CN=MyCompany CA, O=Trust Community, C=DE)

Example: `icm/trusted_reverse_proxy_0 = SUBJECT="CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE", ISSUER="CN=MyCompany CA, O=Trust Community, C=DE"`

Caution

The ICM expects *blanks after the separating comma* for the issuer and subject elements. So, even if the Cloud Connector administration UI shows a string without blanks, for example: `CN=SCC,OU=BTP Scenarios,O=Trust Community,C=DE`, you must specify in ICM: `CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE`.

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

Note

Use the parameters below (instead of `icm/trusted_reverse_proxy_<x>`) only if your kernel release does not yet support parameter `icm/trusted_reverse_proxy_<x>`.

- `icm/HTTPS/trust_client_with_issuer`: Issuer of the system certificate (example data: CN=MyCompany CA, O=Trust Community, C=DE)
- `icm/HTTPS/trust_client_with_subject`: Subject of the system certificate (example data: CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE)

Note

Make sure `icm/HTTPS/verify_client` is set to **1** (request certificate) or **2** (require certificate). If set to **0**, trust cannot be established. The default value is **1**, so it is OK if the parameter is not set at all.

Back to [Tasks \[page 320\]](#)

Back to [Concept \[page 319\]](#)

Next Steps

Now you can proceed with:

- Step 1 of the basic identity propagation setup for HTTPS, see [Configure an ABAP System to Trust the Cloud Connector's System Certificate \[page 315\]](#). However, when using SAP Web Dispatcher, the ABAP backend must trust the *SAP Web Dispatcher* instead of the *Cloud Connector*, see [Forward SSL Certificates for X.509 Authentication](#), step 2 for details.

Then perform the remaining steps of the basic identity propagation setup for HTTPS as described here:

- [Map Short-Lived Certificates to Users \[page 317\]](#)
- [Access ICF Services \[page 317\]](#)

Back to [Tasks \[page 320\]](#)

Back to [Concept \[page 319\]](#)

1.5.2.3.1.3.2 Configure Identity Propagation for RFC

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

Configuring principal propagation for RFC requires an SNC (Secure Network Communications) connection. To enable SNC, you must configure the ABAP system and the Cloud Connector accordingly.

The following example provides step-by-step instructions for the SNC setup.

📘 Note

It is important that you use the same SNC implementation on both communication sides. Contact the vendor of your SNC solution to check the compatibility rules.

Example Data

The following data and setup is used:

📘 Note

The parameters provided in this example are based on an SNC implementation that uses the **SAP Cryptographic Library**. Other vendors' libraries may require different values.

- An SNC identity has been generated and installed on the Cloud Connector host. Generating this identity for the SAP Cryptographic Library is typically done using the tool `SAPGENPSE`. For more information, see [Configuring SNC for SAPCRYPTOLIB Using SAPGENPSE](#).

- The ABAP system is configured properly for SNC.

📘 Note

For the latest system releases, you can use the SSO wizard to configure SNC (transaction code: SNCWIZARD). System prerequisites are described in SAP note [2015966](#) .

- The Cloud Connector system identity's SNC name is **p:CN=SCC, OU=SAP CP Scenarios, O=Trust Community, C=DE**.
- The ABAP system's SNC identity name is **p:CN=SID, O=Trust Community, C=DE**. This value can typically be found in the ABAP system instance profile parameter `snc/identity/as` and hence is provided per application server.
- When using the SAP Cryptographic Library, the ABAP system's SNC identity and the Cloud Connector's system identity should be signed by the same CA for mutual authentication.
- The example short-lived certificate has the subject **CN=P1234567**, where **P1234567** is the SAP BTP application user.

Tasks

1. [Configure the ABAP System \[page 323\]](#)
2. [Map Short-Lived Certificates to Users \[page 324\]](#)
3. [Configure the Cloud Connector \[page 324\]](#)

1. Configure the ABAP System to Trust the Cloud Connector's System SNC identity

1. Open the *SNC Access Control List for Systems* (transaction SNC0).
2. As the Cloud Connector does not have a system ID, use an arbitrary value for `<System ID>` and enter it together with its SNC name: **p:CN=SCC, OU=SAP CP Scenarios, O=Trust Community, C=DE**.
3. Save the entry and choose the *Details* button.
4. In the next screen, activate the checkboxes for *Entry for RFC activated* and *Entry for certificate activated*.
5. Save your settings.

Back to [Tasks \[page 323\]](#)

Back to [Example Data \[page 322\]](#)

2. Map Short-Lived Certificates to Users

You can do this manually in the system as described below or use an identity management solution for a more comfortable approach. For example, for large numbers of users the rule-based certificate mapping is a good way to save time and effort. See [Rule-Based Certificate Mapping](#).

1. Open [Assignment of External ID to Users](#) (transaction `EXTID_DN`).
2. Switch to the edit mode.
3. Create a new entry. Specify the subject of the certificate as `External ID`. Using the example data, this is `CN=P1234567`. In the `<User>` field, provide an appropriate ABAP user, for example `JOHNDOE`.
4. Save the mapping.
5. Repeat the previous steps for all users that should be supported for the scenario.

Back to [Tasks \[page 323\]](#)

Back to [Example Data \[page 322\]](#)

3. Configure the Cloud Connector

[Prerequisites \[page 324\]](#)


[Set up the Cloud Connector to Use the SNC Implementation \[page 324\]](#)

[Create an RFC Hostname Mapping \[page 325\]](#)

Prerequisites

- The required security product for the SNC flavor that is used by your ABAP back-end systems, is installed on the Cloud Connector host.
- The Cloud Connector's system SNC identity is associated with the operating system user under which the Cloud Connector process is running.

Note

If you use SAP Cryptographic Library as SNC implementation, follow the steps described in [Initial Configuration \(RFC\) \[page 287\]](#). Additionally, SAP note [2642538](#)  provides a good description to associate an SNC identity of SAP Cryptographic Library with a user running an external program that uses JCo. When using a different SNC offering, get in touch with the SNC library vendor for details.

Back to [Step \[page 324\]](#)

Set up the Cloud Connector to Use the SNC Implementation

1. In the Cloud Connector UI, choose [Configuration](#) from the main menu, select the [On Premise](#) tab, and go to the [SNC](#) section.
2. Provide the fully qualified name of the SNC library (the security product's shared library implementing the GSS API), the SNC name of the above system identity, and the desired quality of protection by choosing the [Edit](#) icon.
For more information, see [Initial Configuration \(RFC\) \[page 287\]](#).

Note

The example in [Initial Configuration \(RFC\) \[page 287\]](#) shows the library location if you use the SAP Secure Login Client as your SNC security product. In this case (as well as for some other security products), *SNC My Name* is optional, because the security product automatically uses the identity associated with the current operating system user under which the process is running, so you can leave that field empty. (Otherwise, in this example it should be filled with **p:CN=SCC, OU=SAP CP Scenarios, O=Trust Community, C=DE.**)

We recommend that you enter **Maximum Protection** for **<Quality of Protection>**, if your security solution supports it, as it provides the best protection.

3. Choose *Save* and *Close*.

Back to [Step \[page 324\]](#)

Create an RFC Hostname Mapping

1. In the *Access Control* section of the Cloud Connector, create a hostname mapping corresponding to the cloud-side RFC destination. See [Configure Access Control \(RFC\) \[page 350\]](#).
2. Make sure you choose **RFC SNC** as **<Protocol>** and **ABAP System** as **<Back-end Type>**. In the **<SNC Partner Name>** field, enter the ABAP system's SNC identity name, for example, **p:CN=SID, O=Trust Community, C=DE.**
3. Save your mapping.

Back to [Step \[page 324\]](#)

Back to [Tasks \[page 323\]](#)

Back to [Example Data \[page 322\]](#)

Related Information

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

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

[Rule-Based Mapping of Certificates \[page 325\]](#)

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

[Principal Propagation \[page 130\]](#) (Neo environment)

1.5.2.3.1.3.3 Rule-Based Mapping of Certificates

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

Note

The information in this section applies to both principal propagation and technical user propagation.

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

1. Enter a dynamic parameter using transaction RZ11.
 1. Enter the `login/certificate_mapping_rulebased` parameter.
 2. Choose the [Change Value](#) button.
 3. Enter the value **1**.
 4. Save the value.

Note

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

2. Configure rule-based mapping
 1. To create a sample certificate with the Cloud Connector, logon to the Cloud Connector UI and from the main menu, choose [Configuration](#). In the [System Certificate](#) section, enter a sample CN name and download the sample certificate to the [Downloads](#) folder of your machine.
 2. To import the sample certificate into the ABAP server, choose transaction CERTRULE and select [Import certificate](#).

Note

To access transaction CERTRULE, you need the corresponding authorizations (see: [Assign Authorization Objects for Rule-based Mapping \[page 327\]](#)).

3. To create explicit rule mappings, choose the [Rule](#) button.
4. Choose [Save](#).

Note

When you save the changes and return to transaction CERTRULE, the sample certificate which you imported in Step 2b will not be saved. This is just a sample editor view to see the sample certificates and mappings.

Related Information

[Rule-Based Certificate Mapping](#)

1.5.2.3.1.3.3.1 Assign Authorization Objects for Rule-based Mapping

Assign authorizations to access transaction CERTRULE.

To access transaction CERTRULE, you need the following authorizations:

- CC control center: System administration (S_RZL_ADM)
 - Activity 03 grants display authorizations.
 - Activity 01 grants change authorizations.
- User Master Maintenance: User Groups (S_USER_GRP)
 - Activity 03 grants display authorizations.
 - Activity 02 grants change authorizations.
 - Class: enter the names of user groups for which the administrator can maintain explicit mappings.

To assign these authorization objects, proceed as follows:

1. Create a *Single Role* using transaction PFCG.
2. To add the authorization objects **S_RZL_ADM** and **S_USER_GRP**, go to the *Authorizations* tab, choose *Change Authorization data* and select the *Manually* button .
3. To generate the profile, choose *Generate* and save the changes.
4. In the *User* tab, enter the user who should execute the transaction CERTRULE.
5. To match the generated profile to the users, choose *User comparison* .

1.5.2.3.1.4 Configure Subject Patterns for Principal Propagation

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

Note

The information in this section applies to both principal propagation and technical user propagation.

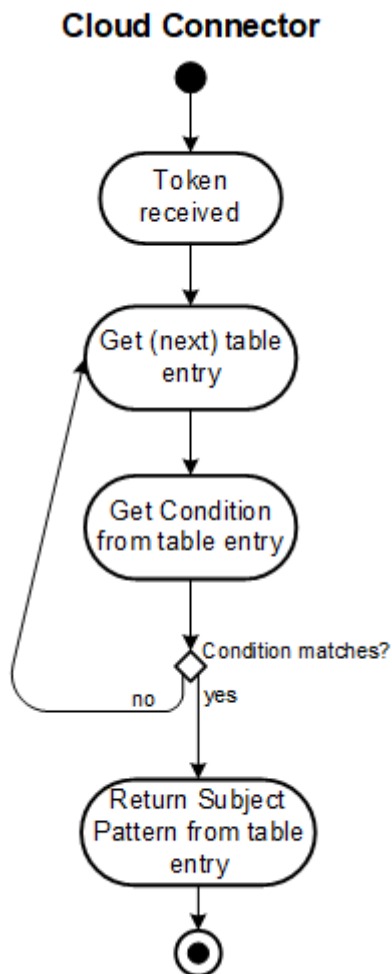
Using this configuration option, you can define different patterns identifying the user for the subject of the generated short-lived X.509 certificate, based on a specified condition. You can also specify the validity period and expiration tolerance.

Configure Subject Patterns

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

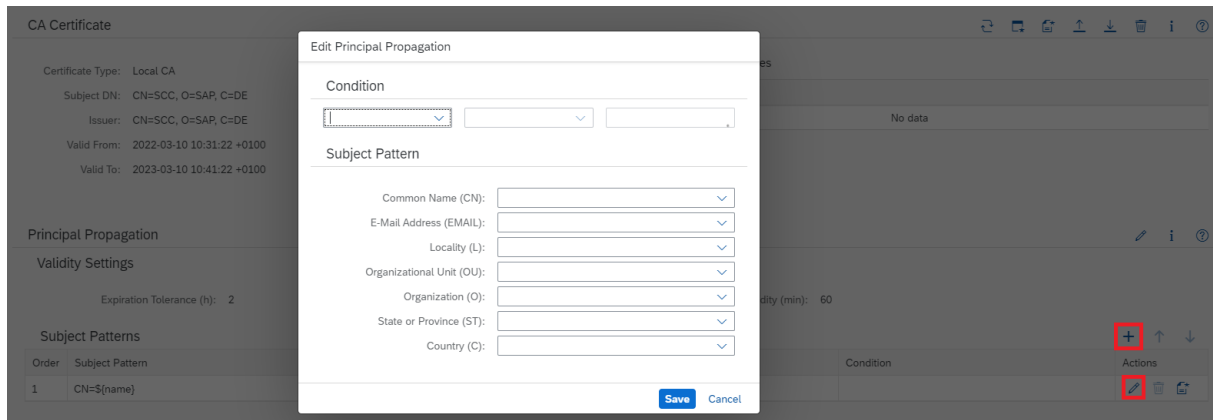
Subject Patterns			+	↑	↓
Order	Subject Pattern	Condition	Actions		
1	CN=\${mail}	\${mail} exists			
2	CN=\${name}				

This table represents an ordered list containing entries that have a specified condition, and the respective subject pattern. You can change the order for an entry by choosing the arrow buttons. The workflow in the Cloud Connector looks like this:



The last entry in the table is the default one having no condition (that is, it always matches as fallback). This entry can not be moved or deleted.

To modify or add table entries, choose the [Edit](#) or [Add](#) icon:



Specify a Condition

Use either of the following procedures to specify a condition based on the attributes of incoming tokens from cloud side:

- Enter the values in the subject pattern fields manually.
- Use the selection menu of the corresponding field to enter a predefined variable.

Using the selection menu, you can assign values for the following parameters:

- `${user_type}`
- `${name}`
- `${mail}`
- `${email}`
- `${display_name}`
- `${login_name}`

Operators

In a next step, choose an operator:

- *exists*: This attribute must be present in the incoming token.
- *does not exist*: This attribute must not be present in the incoming token.
- *is*: This attribute must be present and equals to the value that has been entered in the third field afterwards.
- *is not*: This attribute must be present and is not equal to the value that has been entered in the third field afterwards.

Note

For the condition `${user_type}`, you can only switch between **Technical** or **Business**. The latter refers to the "classical" propagation of business user information, whereas **Technical** is the propagation of a technical user.

Subject Pattern Details

Use either of the following procedures to define the subject's distinguished name (DN), for which the certificate will be issued:

- Enter the values in the subject pattern fields manually.
- Use the selection menu of the corresponding field to enter a predefined variable.

Using the selection menu, you can assign values for the following parameters:

- `${name}`
- `${mail}`
- `${display_name}`
- `${login_name}`

Note

If the token provided by the Identity Provider contains additional values that are stored in attributes with different names, but you still want to use it for the subject pattern, you can edit the variable name to place the corresponding attribute value in the subject accordingly. For example, provide `${email}`, if a SAML assertion uses `email` instead of providing `mail`, or `${user_uuid}` if the attribute `user_uuid` representing the global user ID is contained in the assertion.

When using a subaccount in the **Cloud Foundry** environment: 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.

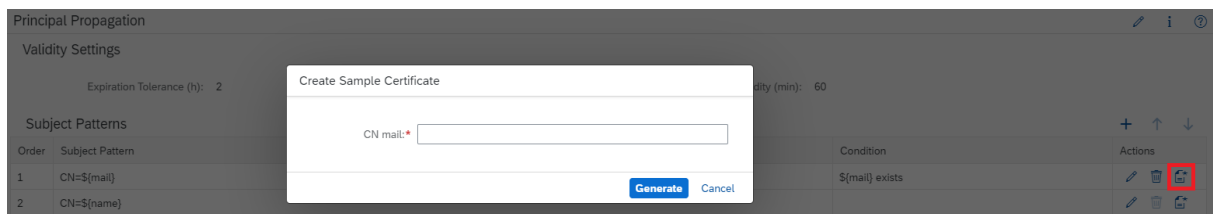
By default, the following attributes are provided:

- `<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.

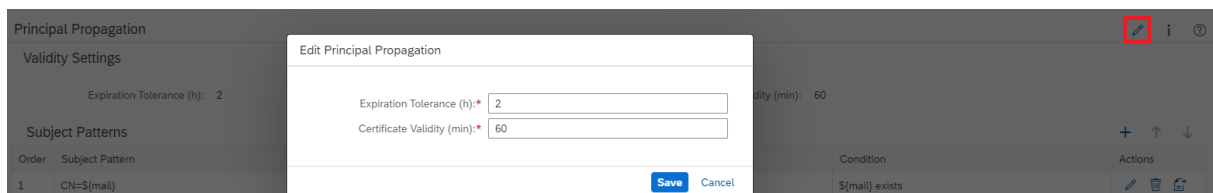
Sample Certificate

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.



Validity Settings

You can change the validity settings by choosing the [Edit](#) button.



Related Information

[Server Certificate Authentication](#)

1.5.2.3.1.5 Configure a Secure Login Server

Configuration steps for Java SLS support.

Note

The information in this section applies to both principal propagation and technical user propagation.

Note

The Secure Login Server mainstream maintenance ends on December 31, 2027.

Content

[Overview \[page 332\]](#)

[Requirements \[page 333\]](#)

[Implementation \[page 333\]](#)

Overview

The Cloud Connector can use on-the-fly generated X.509 user certificates to log in to on-premise systems if the external user session is authenticated (for example by means of SAML). If you do not want to use the built-in certification authority (CA) functionality of the Cloud Connector (for example because of security considerations), you can connect SAP SSO 2.0 Secure Login Server (SLS) or higher.

Note

Make sure you use a version that is still supported, which is currently at least SAP SSO 3.0 Secure Login Server.

SLS is a Java application running on AS JAVA 7.20 or higher, which provides interfaces for certificate enrollment.

SLS supports the following formats:

- HTTPS
- REST
- JSON
- PKCS#10/PKCS#7

Note

Any enrollment requires a successful user or client authentication, which can be a single, multiple or even a multi factor authentication.

The following schemes are supported:

- LDAP/ADS
- RADIUS
- SAP SSO OTP
- ABAP RFC
- Kerberos/SPNego
- X.509 TLS Client Authentication

SLS lets you define arbitrary enrollment profiles, each with a unique profile UID in its URL, and with a configurable authentication and certificate generation.

Back to [Content \[page 332\]](#)

Requirements

For user certification, SLS must provide a profile that adheres to the following:

- Cloud Connector client authentication by its X.509 system certificate
- Cloud Connector system certificate and SLS may live in different PKIs
- Cloud Connector hands over the full user's certificate subject name

With SAP SSO 2.0 SP06, SLS provides the following required features:

- TLS Client Authentication-based enrollment with `SecureLoginModuleUserDelegationWithSSL` (available since SP04)
- multi-PKI support is implemented by all standard components of Application Server (AS) JAVA, AS ABAP, HANA, by importing trusted root CA certificates
- SLS allows `PKCS10:SUBJECT` in a profile's certificate configuration (SP06)

Back to [Content \[page 332\]](#)

Implementation

INSTALLATION

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

Note

SLS allows you to set up one or more own PKIs with Root CA, User CA, and so on. You can also import CAs as PKCS#12 file or use a hardware security module (HSM) as "External User CA".

Note

You should only use HTTPS connections for any communication with SLS. AS JAVA / ICM supports TLS, and the default configuration comes with a self-signed sever certificate. You may use SLS to replace this certificate by a PKI certificate.

CONFIGURATION

SSL Ports

1. Open the NetWeaver Administrator, choose **Configuration > SSL** and define a new port with **Client Authentication Mode = REQUIRED**.

Note

You may also define another port with **Client Authentication Mode = Do not request** if you did not do so yet.

2. Import the root CA of the PKI that issued your Cloud Connector system certificate.
3. Save the configuration and restart the Internet Communication Manager (ICM).

Authentication Policy

1. Open the NetWeaver Administrator (NWA, <https://<host:port>/nwa>).
2. From the top-level menu, choose **Configuration > Authentication and Single Sign-On**.
3. In the **Policy Configuration** table, switch to **Type = Custom**.
4. Choose **Add** to create a new policy and assign it a name, for example, **SecureLoginCloudConnector**.
5. Open **Edit** mode.
6. In the **Details** section of the authentication configuration, choose **Authentication Stack > Login Modules** and add **SecureLoginModuleUserDelegationWithSSL**.
7. In **<Rule1.subjectName>** and **<Rule1.issuerName>**, enter the respective certificate names of your Cloud Connector system certificate.
8. In the **Details** section of the authentication configuration, choose **Properties** and add the property **UserNameMapping** with value **VirtualUser**.
9. Save the policy.

Client Authentication Profile

1. Open the SLS Administration Console (SLAC, <https://host:port/slac>).
2. From the top-level menu, choose **Profile Management > Authentication Profiles**.
3. Create a new profile with **Client Type = Secure Login Client** and assign it a name, for example, **Cloud Connector User Certificates**.
4. Choose **User Authentication > Use Policy Configuration** and select **Policy Configuration Name = SecureLoginCloudConnector**.
5. Edit all required fields in the wizard according to your requirements.

6. In tab [Authentication Configuration](#), check the box *Virtual User*.
7. Save your entries.
8. Select the new profile and open [Edit](#) mode.
9. Choose ► [Certificate Configuration](#) ► [Certificate Name and Alternative Names](#) ► and set `Appendix Subject Name = (PKCS10:SUBJECT)`.
10. Leave all other fields in [Certificate Name and Alternative Names](#) empty.
11. On the [Enrollment Configuration](#) page, make sure that the `<Enrollment URL>` has the correct value, otherwise edit and fix it:
 1. full DNS name
 2. port *with* TLS Client Authentication (see port number in NWA SSL Configuration).
12. Save your entries.

User Profile Group

1. Open the SLS Administration Console (SLAC, <https://host:port/slac>).
2. Go to the top level menu and choose ► [Profile Management](#) ► [User Profile Groups](#) ►.
3. Create a new profile group, make sure the `<Policy URL>` has the correct value:
 1. Full DNS name
 2. Port *without* TLS Client Authentication (see port number in NWA SSL Configuration).
4. In tab [Profiles](#), add the profile `Cloud Connector User Certificates`.
5. Save your entries.

Root CA Certificate

1. Open SLS Administration Console (SLAC, <https://host:port/slac>).
2. Go to the top level menu and choose [Certificate Management](#).
3. Select the Root CA certificate you are using in your profile.
4. Choose ► [Export entry](#) ► [X.509 Certificate](#) ► and download the certificate file.

Cloud Connector

Follow the standard installation procedure of the Cloud Connector and configure SLS support as explained in [Configuration of a CA hosted by a Secure Login Server](#):

1. Enter the policy URL that points to the SLS user profile group.
2. Select the profile, for example, `Cloud Connector User Certificates`.
3. Import the Root CA certificate of SLS into the Cloud Connector's [Trust Store](#).

On-Premise Target Systems

Follow the standard configuration procedure for Cloud Connector support in the corresponding target system and configure SLS support.

To do so, import the Root CA certificate of SLS into the system's truststore:

- **AS ABAP:** choose transaction STRUST and follow the steps in [Maintaining the SSL Server PSE's Certificate List](#).
- **AS Java:** open the Netweaver Administrator and follow the steps described in [Configuring the SSL Key Pair and Trusted X.509 Certificates](#).

Back to [Content \[page 332\]](#)

1.5.2.3.1.6 Configure Kerberos

Context

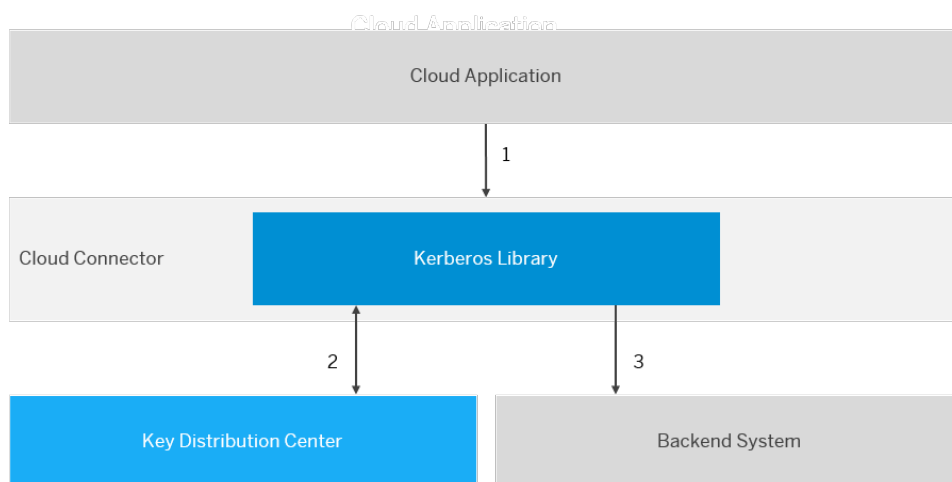
The Cloud Connector allows you to propagate users authenticated in SAP BTP via Kerberos against backend systems. It uses the **Service For User and Constrained Delegation** protocol extension of Kerberos.

Note

This feature is not supported for ABAP backend systems. In this case, you can use the certificate-based principal propagation, see [Configure a CA Certificate \[page 308\]](#).

The Key Distribution Center (KDC) is used for exchanging messages in order to retrieve Kerberos tokens for a certain user and backend system.

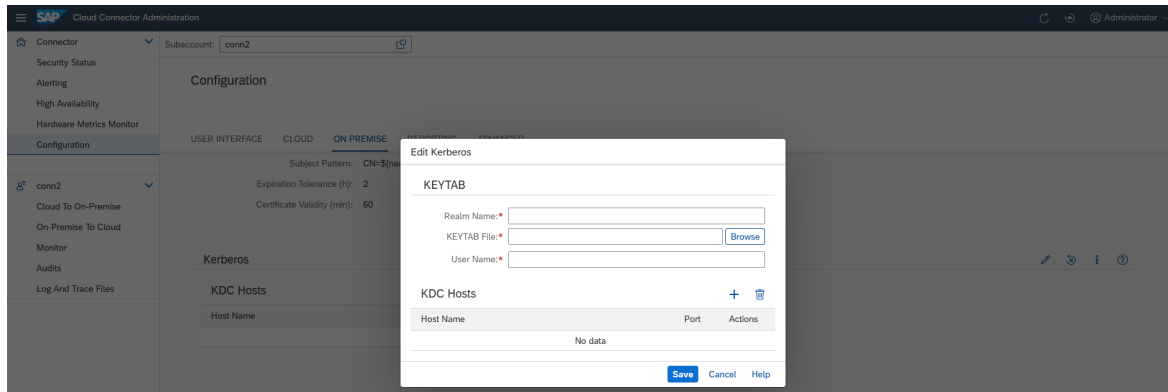
For more information, see [Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol](#).



1. An SAP BTP application calls a backend system via the Cloud Connector.
2. The Cloud Connector calls the KDC to obtain a Kerberos token for the user propagated from the Cloud Connector.
3. The obtained Kerberos token is sent as a credential to the backend system.

Procedure

1. Choose [Configuration](#) from the main menu.
2. From the [Kerberos](#) section of the [On Premise](#) tab, choose [Edit](#).



3. Enter the name of your Kerberos realm.
4. Upload a KEYTAB file that contains the secret keys of your service user. The KEYTAB file should contain the **rc4-hmac** key for your user.
5. Enter the name of the service user to be used for communication with the KDC. This user should be allowed to request Kerberos tokens for other users for the backend systems that you are going to access.
6. In the **<KDC Hosts>** field (press **Add** to display the field), enter the host name of your KDC using the format **<host>:<port>**. The port is optional; if you leave it empty, the default, 88, is used.
7. Choose **Save**.

Example

You have a backend system protected with SPNego authentication in your corporate network. You want to call it from a cloud application while preserving the identity of a cloud-authenticated user.

Define the following:

- A connectivity destination in SAP BTP, with `ProxyType` = **OnPremise**.
- A system mapping made in the Cloud Connector. (Choose **Cloud to On Premise** from your subaccount menu, Go to tab **Access Control** **Add**, and for **Principal Type**, select **kerberos**.)
- Kerberos configuration in the Cloud Connector, where the service user is allowed to delegate calls for your backend host service.

Result:

When you now call a backend system, the Cloud Connector obtains an SPNego token from your KDC for the cloud-authenticated user. This token is sent along with the request to the back end, so that it can authenticate the user and the identity to be preserved.

Related Information

[Set Up Trust \[page 304\]](#)

1.5.2.3.17 Configuring Identity Propagation to SAP NetWeaver AS for Java

Find step-by-step instructions on how to set up an application server for Java (AS Java) to enable identity propagation for HTTPS.

Note

The information in this section applies to both principal propagation and technical user propagation. These two types of user propagation are combined as *identity* propagation.

Prerequisites

To perform the following steps, you must have the corresponding administrator authorizations in AS Java (SAP NetWeaver Administrator) as well as an administrator user for the Cloud Connector.

Configure AS Java to Trust the Cloud Connector's System Certificate

Procedure

1. Go to ► [SAP NetWeaver Administrator](#) ► [Certificates and Keys](#) ► and import the Cloud Connector's system certificate into the *Trusted CAs* keystore view. See [Importing Certificate and Key From the File System](#).
2. Configure the Internet Communication Manager (ICM) to trust the system certificate for identity propagation.
 - a. Add a new SSL access point. See [Adding New SSL Access Points](#).
 - b. Generate a certificate signing request and send it to the CA of your choice. See [Configuration of the AS Java Keystore Views for SSL](#).
 - c. Import the certificates and save the configuration.

Import the certificate signing response, the root X.509 certificate of the trusted CA, and the Cloud Connector's system certificate into the new SSL access point from step 2a. Save the configuration and restart the ICM. See [Configuring the SSL Key Pair and Trusted X.509 Certificates](#).
 - d. Make sure the ICM trusts the system certificate for identity propagation. For more information, see [Using Client Certificates via an Intermediary Server](#).
 - e. Restart the ICM and test the SSL connection. For more information, see [Testing the SSL Connection](#).

Define Rules for User Mapping

Procedure

1. Add the [ClientCertLoginModule](#) to the policy configuration that the Cloud Connector connects to. See [Configuring the Login Module on the AS Java](#).
2. Define the rules to map users authenticated with their certificate to users that exist in the User Management Engine. See [Using Rules for User Mapping in Client Certificate Login Module](#).
 - To map the user ID of the certificate's subject name field to users, see [Using Rules Based on Client Certificate Subject Names](#).
 - To map the user ID based on rules for the certificate V3 extension *SubjectAlternativeName*, see [Using Rules Based on Client Certificate V3 Extensions](#).
 - To use client certificate filters, see [Defining Rules for Filtering Client Certificates](#).

Related Information

[Configuring Transport Layer Security on SAP NetWeaver AS for Java Using X.509 Client Certificates](#)
[Configure Identity Propagation for HTTPS \[page 314\]](#)

1.5.2.3.2 Configuring Technical User Propagation

Use technical user propagation to provide access of technical users to on-premise systems.

Using technical user propagation, you can forward the identity of technical users that are identified using an OAuth client and act on behalf of this identity in an on-premise system.

Note

Technical user propagation is possible only in the Cloud Foundry environment.

Tasks in this section:

Task	Description
Set Up Trust [page 304]	Configure a trusted relationship in the Cloud Connector to support technical user propagation. Technical user propagation lets you forward a technical user in the cloud to the internal system without requesting a password.

Task	Description
Configure a CA Certificate [page 308]	Install and configure an X.509 certificate to enable support for technical user propagation.
Configuring Identity Propagation to an ABAP System [page 313]	Learn more about the different types of configuring and supporting technical user propagation for a particular AS ABAP.
Configure Subject Patterns for Principal Propagation [page 327]	Define a pattern identifying the user for the subject of the generated short-lived X.509 certificate, as well as its validity period.
Configure a Secure Login Server [page 332]	Configuration steps for Java Secure Login Server (SLS) support.
Configure Kerberos [page 336]	The Cloud Connector lets you propagate users authenticated in SAP BTP via Kerberos against backend systems. It uses the Service For User and Constrained Delegation protocol extension of Kerberos.
Configuring Identity Propagation to SAP NetWeaver AS for Java [page 338]	Find step-by-step instructions on how to configure technical user propagation to an application server Java (AS Java).

Related Information

[OAuth Technical User Propagation Authentication](#)

1.5.2.4 Configure Access Control

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

To allow your cloud applications to access a certain backend system on the intranet, you must specify this system in the Cloud Connector. The procedure is specific to the protocol that you are using for communication.

Find the detailed configuration steps for each communication protocol here:

[Configure Access Control \(HTTP\) \[page 342\]](#)

[Configure Access Control \(RFC\) \[page 350\]](#)

[Configure Access Control \(LDAP\) \[page 357\]](#)

[Configure Access Control \(TCP\) \[page 359\]](#)

Copy Access Control Settings

When you add new subaccounts, you can copy the complete access control settings from another subaccount on the same Cloud Connector. You can also do it any time later by using the import/export mechanism provided by the Cloud Connector.

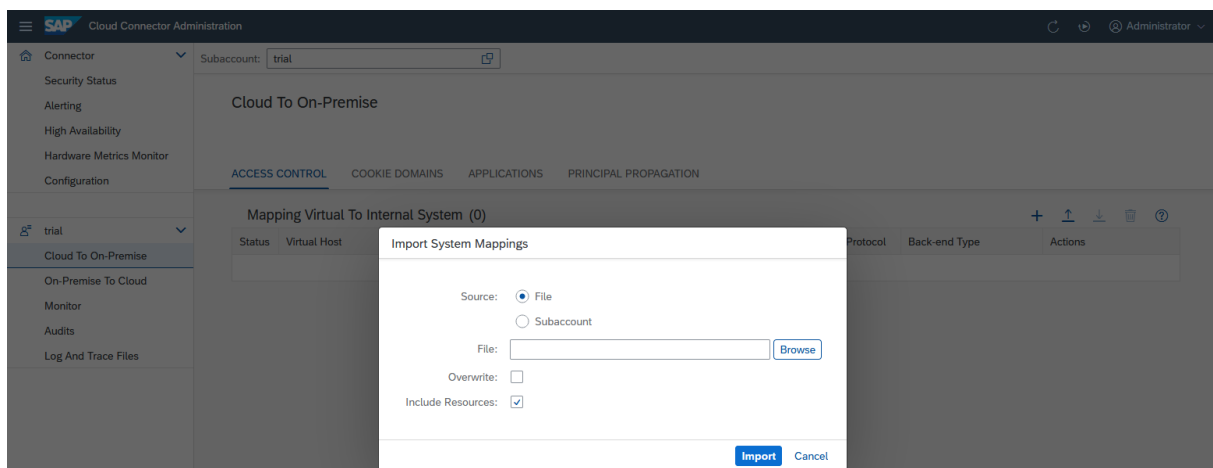
Export Access Control Settings

1. From your subaccount menu, choose [Cloud To On-Premise](#) and select the tab [Access Control](#).
2. To store the current settings in a ZIP file, choose [Download](#) icon in the upper-right corner.
3. You can later import this file into a different Cloud Connector.

Import Access Control Settings

There are two locations from which you can import access control settings:

- A file that was previously exported from a Cloud Connector
- A different subaccount on the same Cloud Connector



Two additional options define the behavior of the import:

- **Overwrite:** Select this checkbox if you want to replace existing system mappings with imported ones. Do not select this checkbox if you want to keep existing mappings and only import the ones that are not yet available (default).

Note

A system mapping is uniquely identified by the combination of virtual host and port.

- **Include Resources:** When this checkbox is selected (default), the resources that belong to an imported system are also imported. Otherwise no resources are imported, that is, imported system mappings do not expose any resources.

Related Information

[Configure Access Control \(HTTP\) \[page 342\]](#)

[Configure Access Control \(RFC\) \[page 350\]](#)

[Configure Access Control \(LDAP\) \[page 357\]](#)

[Configure Access Control \(TCP\) \[page 359\]](#)

[Configure Accessible Resources \[page 362\]](#)

[Configure Domain Mappings for Cookies \[page 504\]](#)

1.5.2.4.1 Configure Access Control (HTTP)

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

To allow your cloud applications to access a certain backend system on the intranet via HTTP, you must specify this system in the Cloud Connector.

Note

Make sure that also redirect locations are configured as internal hosts.

If the target server responds with a redirect HTTP status code (30x), the cloud-side HTTP client usually sends the redirect over the Cloud Connector as well. The Cloud Connector runtime then performs a reverse lookup to rewrite the location header that indicates where to route the redirected request.

If the redirect location is ambiguous (that is, several mappings point to the same internal host and port), the first one found is used. If none is found, the location header stays untouched.

Tasks

[Expose Intranet Systems \[page 343\]](#)

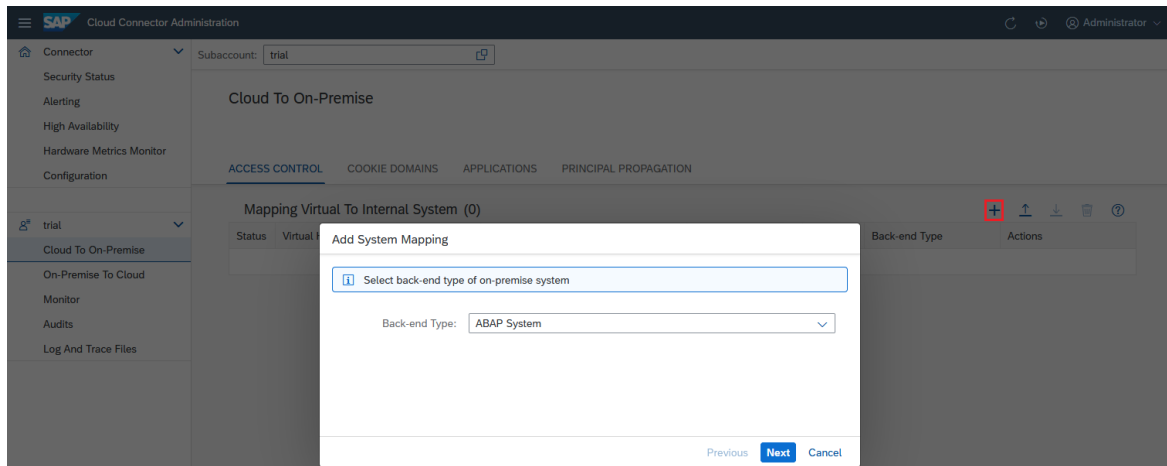
[Limit the Accessible Services for HTTP\(S\) \[page 348\]](#)

[Activate or Suspend Resources \[page 349\]](#)

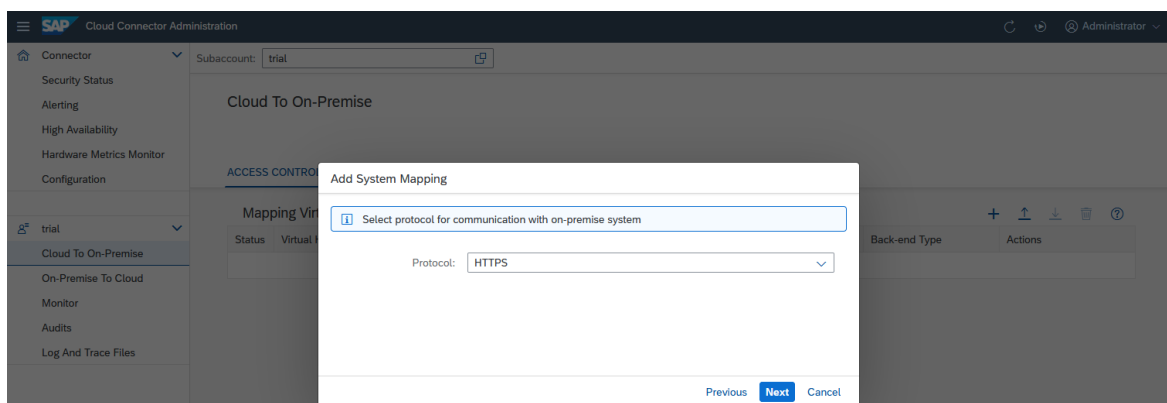
Expose Intranet Systems

Insert a new entry in the Cloud Connector access control management:

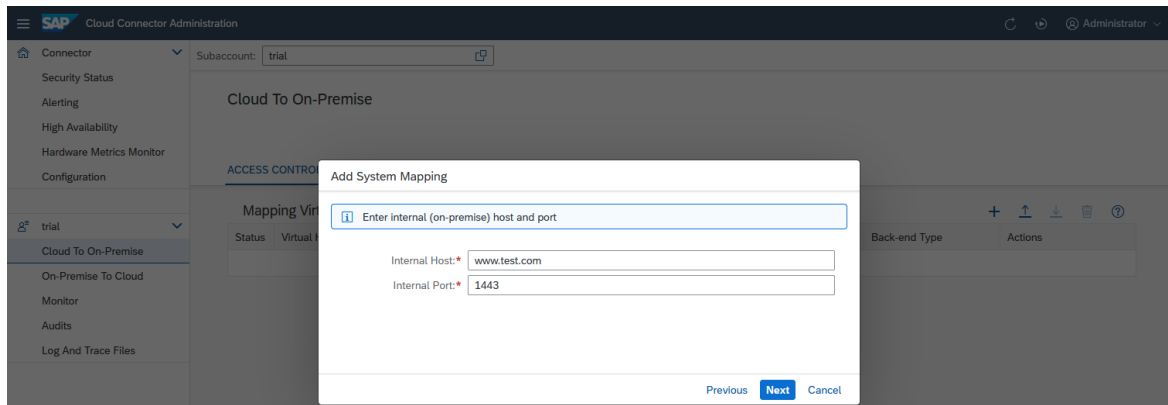
1. Choose *Cloud To On Premise* from your *Subaccount* menu.
2. Choose *Add*. A wizard will open and ask for the required values.
3. *Backend Type*: Select the description that matches best the addressed backend system.
When you are done, choose *Next*.



4. *Protocol*: This field allows you to decide whether the Cloud Connector should use **HTTP** or **HTTPS** for the connection to the backend system. Note that this is completely independent from the setting on cloud side. Thus, even if the HTTP destination on cloud side specifies "http://" in its URL, you can select **HTTPS**. This way, you are ensured that the entire connection from the cloud application to the actual backend system (provided through the TLS tunnel) is TLS-encrypted. The only prerequisite is that the backend system supports HTTPS on that port. For more information, see [Initial Configuration \(HTTP\)](#) [page 285].
 - If you specify HTTPS and there is a "system certificate" imported in the Cloud Connector, the latter attempts to use that certificate for performing a client-certificate-based logon to the backend system.
 - If there is no system certificate imported, the Cloud Connector opens an HTTPS connection without client certificate.



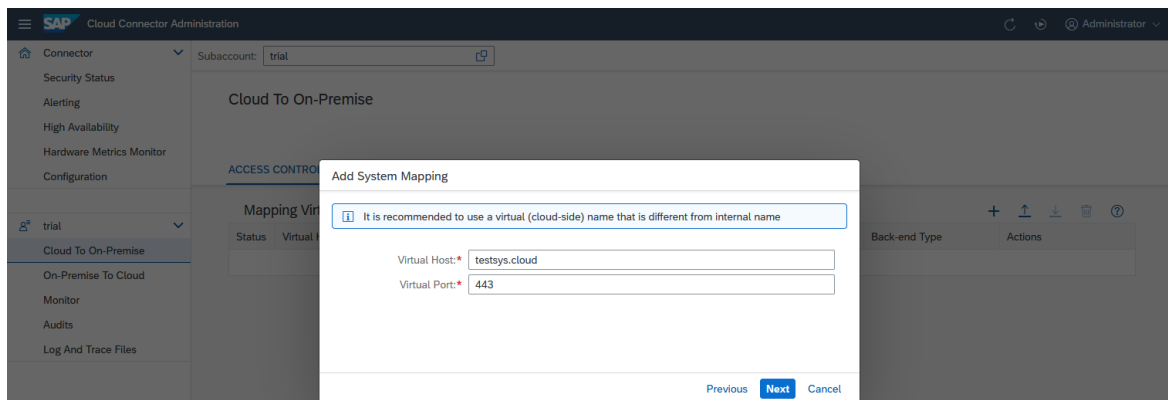
5. *Internal Host* and *Internal Port* specify the actual host and port under which the target system can be reached within the intranet. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector without any proxy. Cloud Connector will try to forward the request to the network address specified by the internal host and port, so this address needs to be real.



6. *Virtual Host* identifies the host name exactly as specified in the `<URL>` property of the HTTP destination configuration in SAP BTP.

See: [Create HTTP Destinations \[page 79\]](#) (Neo environment)

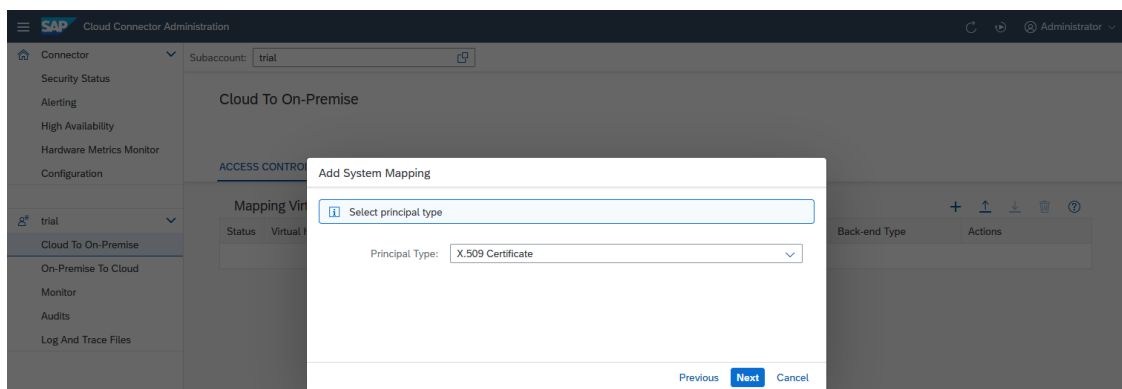
The virtual host can be a fake name and does not need to exist. The *Virtual Port* allows you to distinguish between different entry points of your backend system, for example, **HTTP/80** and **HTTPS/443**, and to have different sets of access control settings for them. For example, some non-critical resources may be accessed by HTTP, while some other critical resources are to be called using HTTPS only. The fields are prepopulated with the values of the *Internal Host* and *Internal Port*. If you don't modify them, you must provide your internal host and port also in the cloud-side destination configuration or in the URL used for your favorite HTTP client.



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

8. *Principal Type*

- Procedure for Cloud Connector version 2.15 and higher:**
 Defines what kind of principal is sent to the backend within the HTTP request. For the principal type **X.509 Certificate** (if a principal is sent from the cloud side), the principal is injected as an HTTP header (`SSL_CLIENT_CERT`) and forwarded to the backend. There, depending on the backed configuration, it can be used for authentication.



- Procedure for Cloud Connector versions below 2.15:**
 You can use two different variants of an X.509 certificate to define the principal type that is sent to the backend within the HTTP request: **X.509 Certificate (General Usage)** or **X.509 Certificate (Strict Usage)**. The latter was introduced with Cloud Connector 2.11.
 If a principal is sent from the cloud side, it is injected in both cases as an HTTP header (`SSL_CLIENT_CERT`) and forwarded to the backend. If the backend is configured correctly for principal propagation, this certificate can be used for authentication.
 However, if the cloud side does not send a principal, the variants behave differently:
 - General Usage** (as well as `<Principal Type> = None`) allows to alternatively use the system certificate for the *TLS handshake* (actually used for trust) also for authentication.
 - Strict Usage** does not allow this. In this case, another authentication type (specified in an additional header) is used instead, for example, basic authentication.
 This setting also applies to HTTP authentication types other than principal propagation.

Note

The recommended variant is **X.509 Certificate (Strict Usage)** as this lets you use principal propagation and, for example, basic authentication over the same access control entry, regardless of the logon order settings in the target system.

To prevent the use of principal propagation to the target system, choose **None** as **<Principal Type>**. In this case, no principal is injected.

For more information on principal propagation, see [Configuring Principal Propagation \[page 304\]](#).

9. **System Certificate for Logon** (available as of Cloud Connector 2.15): Specifies if the Cloud Connector's system certificate should be used for authentication at the backend, if

1. No principal is received, or
2. Principal propagation is not allowed over this mapping at all.

If *activated*, the system certificate of the TLS handshake used for trust is also used for authentication.

If *not activated*, an additional HTTP header (SSL_CLIENT_CERT) is sent. It indicates to the target system that the system certificate used for trust must not be used for authentication.

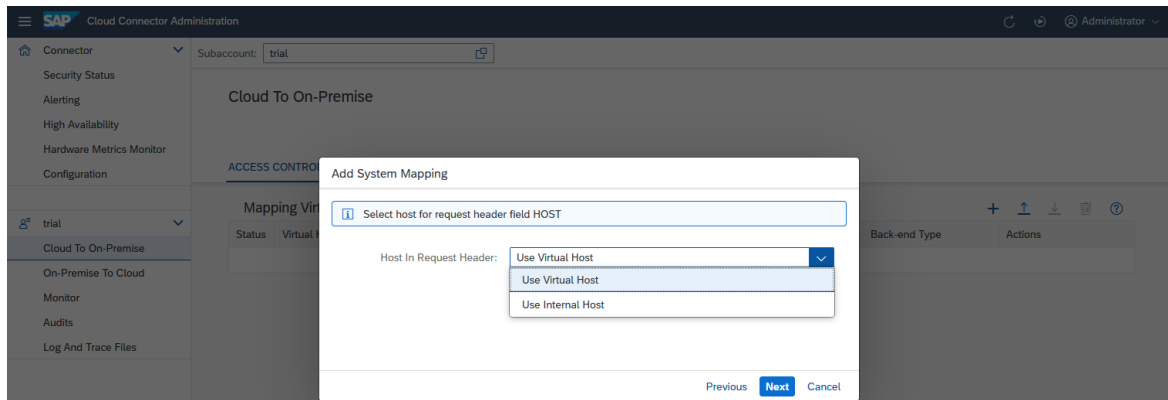
While unselected, another authentication method is used, for example, basic authentication.

Note

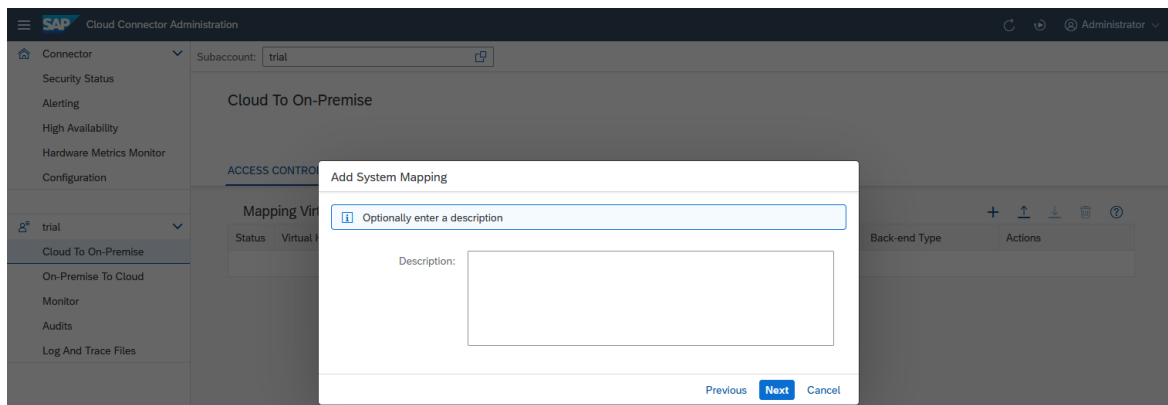
We recommend that you **keep this option deactivated**, as this lets you use principal propagation and basic authentication over the same access control entry, regardless of the logon order settings in the target system.

The screenshot shows a dialog box titled "Add System Mapping". Inside, there is a light blue information box with an 'i' icon and the text: "Choose whether the System Certificate will be used for logon in case no Principal is received from Cloud". Below this, the text "System Certificate for Logon:" is followed by an unchecked checkbox. At the bottom right of the dialog, there are three buttons: "Previous" (disabled), "Next" (active/highlighted), and "Cancel" (disabled).

10. **Host In Request Header** lets you define, which host is used in the host header that is sent to the target server. By choosing **Use Internal Host**, the actual host name is used. When choosing **Use Virtual Host**, the virtual host is used. In the first case, the virtual host is still sent via the X-Forwarded-Host header.




11. You can enter an optional description at this stage. The respective description will be shown when pressing the [Info](#) button of the access control entry (table [Mapping Virtual to Internal System](#)).

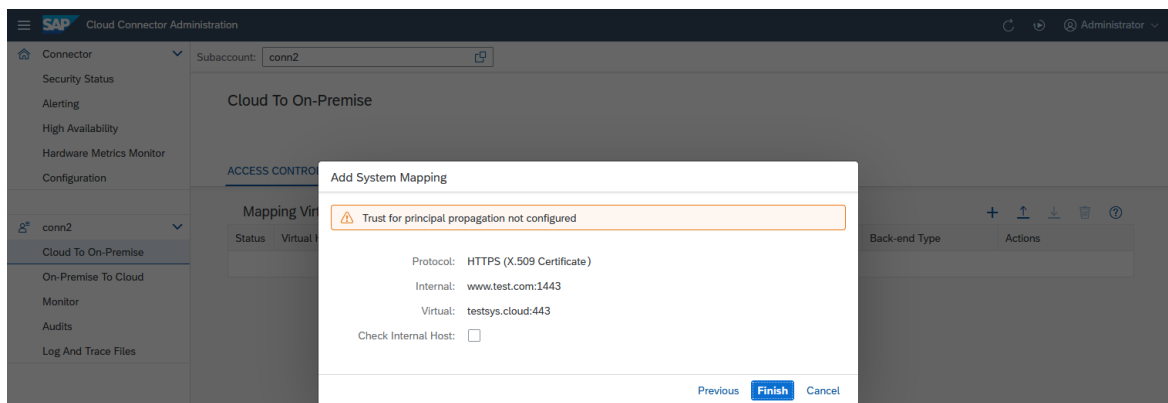


12. The summary shows information about the system to be stored and when saving the host mapping, you can trigger a ping from the Cloud Connector to the internal host, using the [Check availability of internal host](#) checkbox. This allows you to make sure the Cloud Connector can indeed access the internal system, and allows you to catch basic things, such as spelling mistakes or firewall problems between the Cloud Connector and the internal host.

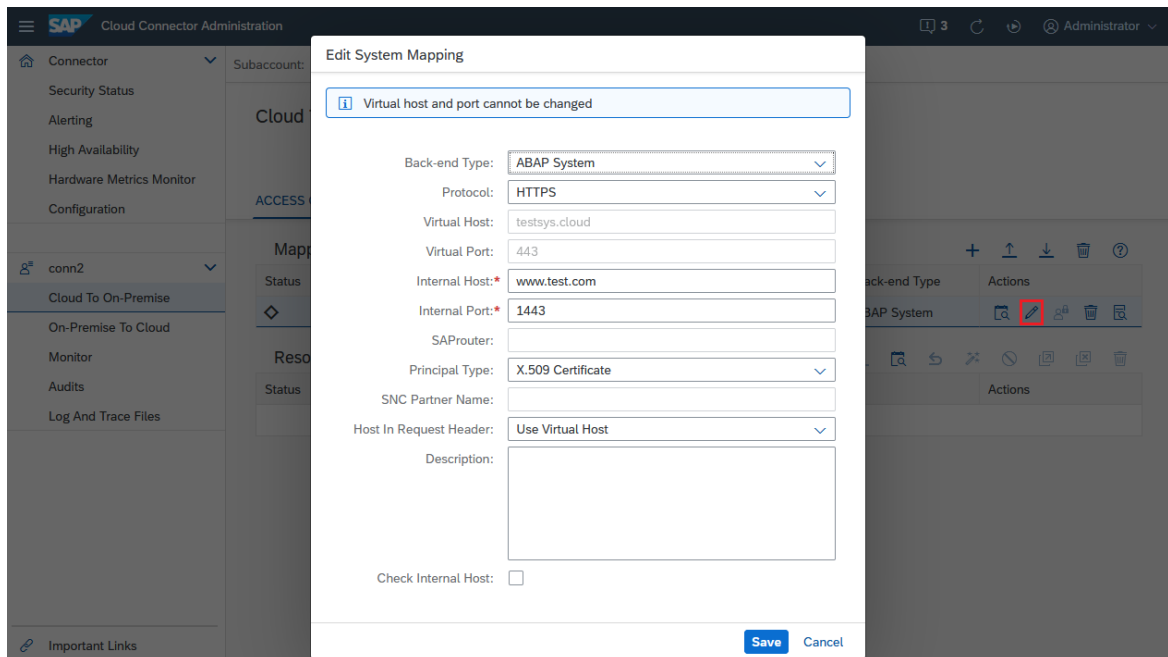
If the ping to the internal host is successful (that is, the host is reachable via TLS), the state `Reachable` is shown. If it fails, a warning pops up. You can view issue details by choosing the [Details](#) button, or check them in the log files.

This check also tries to perform client authentication if possible, regardless of the host's availability. Find additional information and hints by choosing the [Details](#) button. You can check, for example, if the system certificate acting as a client certificate is configured correctly, and if the ABAP backend trusts it.

You can execute the availability check for all selected systems in the [Access Control](#) overview by pressing the button  ([Check availability...](#)) in column [Actions](#).



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.

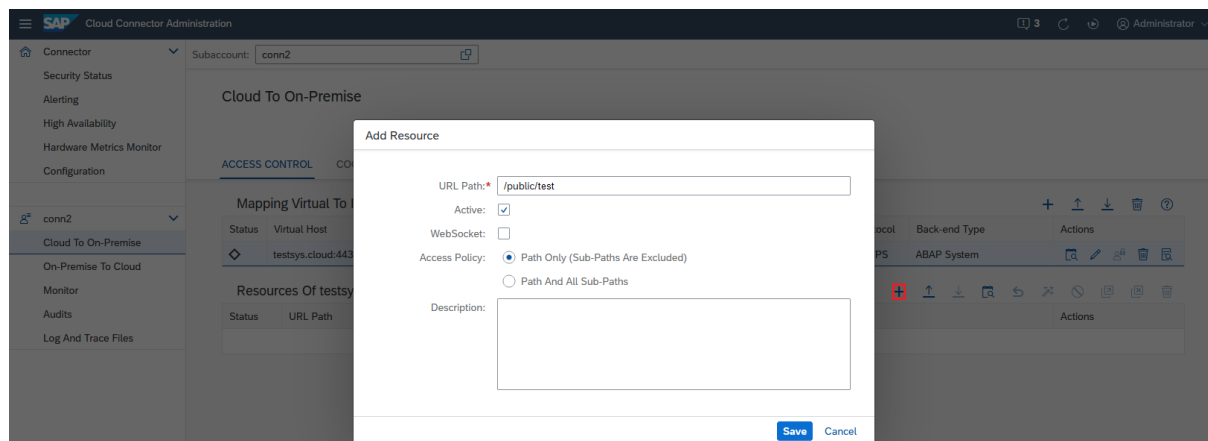


Back to [Tasks \[page 342\]](#)

Limit the Accessible Services for HTTP(S)

In addition to allowing access to a particular host and port, you also must specify which URL paths ([Resources](#)) are allowed to be invoked on that host. The Cloud Connector uses very strict allowlists for its access control. Only those URLs for which you explicitly granted access are allowed. All other HTTP(S) requests are denied by the Cloud Connector.

To define the permitted URLs for a particular backend system, choose the line corresponding to that backend system and choose [Add](#) in section [Resources Accessible On...](#) below. A dialog appears prompting you to enter the specific URL path that you want to allow to be invoked.



The Cloud Connector checks that the path part of the URL (up to but not including a possible question mark (?) that may denote the start of optional CGI-style query parameters) is exactly as specified in the configuration. If it is not, the request is denied. If you select option [Path and all sub-paths](#), the Cloud Connector allows all requests for which the URL path (not considering any query parameters) starts with the specified string.

The [Active](#) checkbox lets you specify, if that resource is initially enabled or disabled. See the section below for more information on enabled and disabled resources.

The [WebSocket Upgrade](#) checkbox lets you specify, whether that resource allows a protocol upgrade.

Back to [Tasks \[page 342\]](#)

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
	/public/test	Path Only (Sub-Paths Are Excluded)	

- To activate the resource again, select it and choose the [Activate](#) button.
- By choosing [Allow WebSocket upgrade](#)/[Disallow WebSocket upgrade](#) this is possible for the protocol upgrade setting as well.

- It is also possible to mark multiple lines and then suspend or activate all of them in one go by clicking the [Activate/Suspend](#) icons in the top row. The same is true for the corresponding [Allow WebSocket upgrade/Disallow WebSocket](#) icons.

Examples:

- [/production/accounting](#) and [Path only \(sub-paths are excluded\)](#) are selected. Only requests of the form GET [/production/accounting](#) or GET [/production/accounting?name1=value1&name2=value2...](#) are allowed. (GET can also be replaced by POST, PUT, DELETE, and so on.)
- [/production/accounting](#) and [Path and all sub-paths](#) are selected. All requests of the form GET [/production/accounting-plus-some-more-stuff-here?name1=value1...](#) are allowed.
- [/](#) and [Path and all sub-paths](#) are selected. All requests to this server are allowed.

Back to [Tasks \[page 342\]](#)

Related Information

[Configure Domain Mappings for Cookies \[page 504\]](#)

[Consume Backend Systems \(Java Web or Java EE 6 Web Profile\) \[page 171\]](#)

1.5.2.4.2 Configure Access Control (RFC)

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

Tasks

[Expose Intranet Systems \[page 350\]](#)

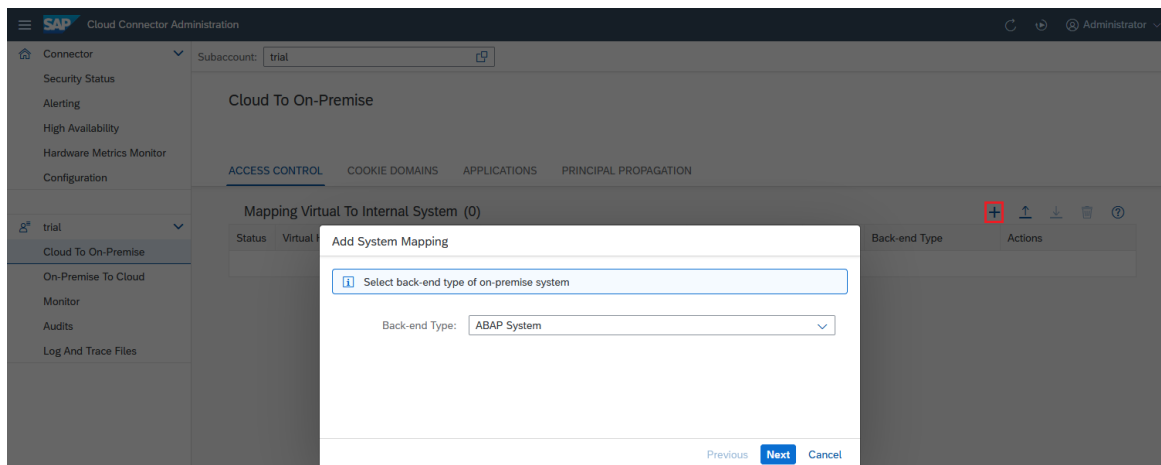
[Limit the Accessible Resources for RFC \[page 356\]](#)

Expose Intranet Systems

To allow your cloud applications to access a certain backend system on the intranet, insert a new entry in the Cloud Connector [Access Control](#) management.

1. Choose [Cloud To On-Premise](#) from your [Subaccount](#) menu and go to tab [Access Control](#).
2. Choose [Add](#).

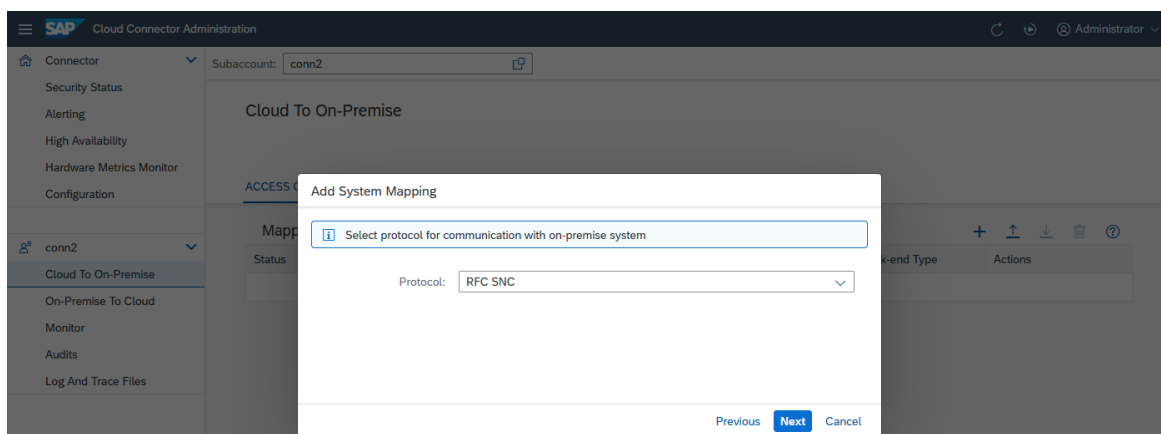
3. **Backend Type**: Select the backend system type (**ABAP System** or **SAP Gateway** for RFC).



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

Note

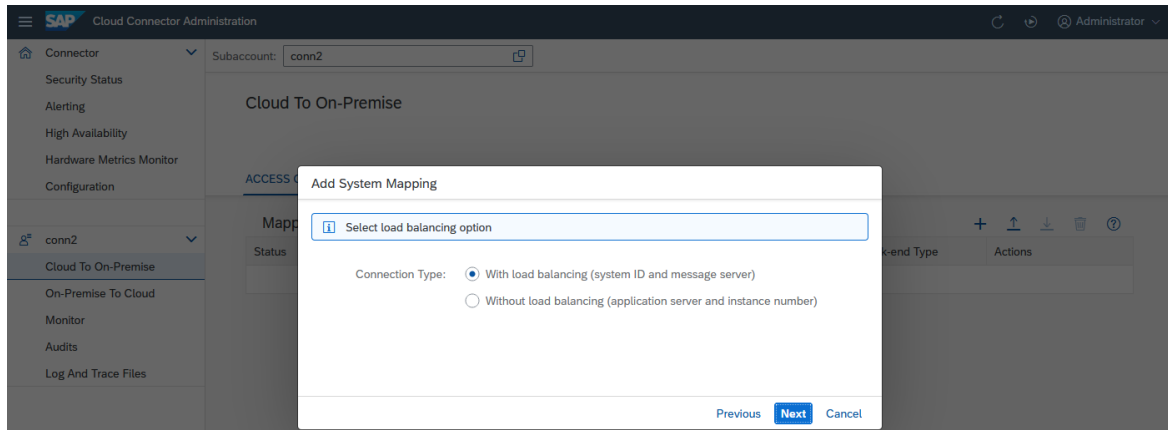
The value **RFC SNC** is independent from your settings on the cloud side, since it only specifies the communication between Cloud Connector and backend system. Using **RFC SNC**, you can ensure that the entire connection from the cloud application to the actual backend system (provided by the SSL tunnel) is secured, partly with SSL and partly with SNC. For more information, see [Initial Configuration \(RFC\)](#) [page 287].



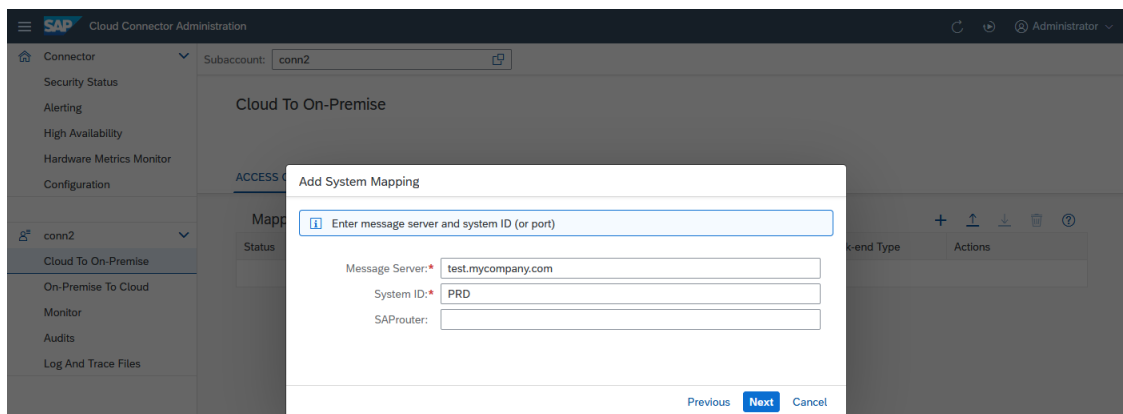
Note

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

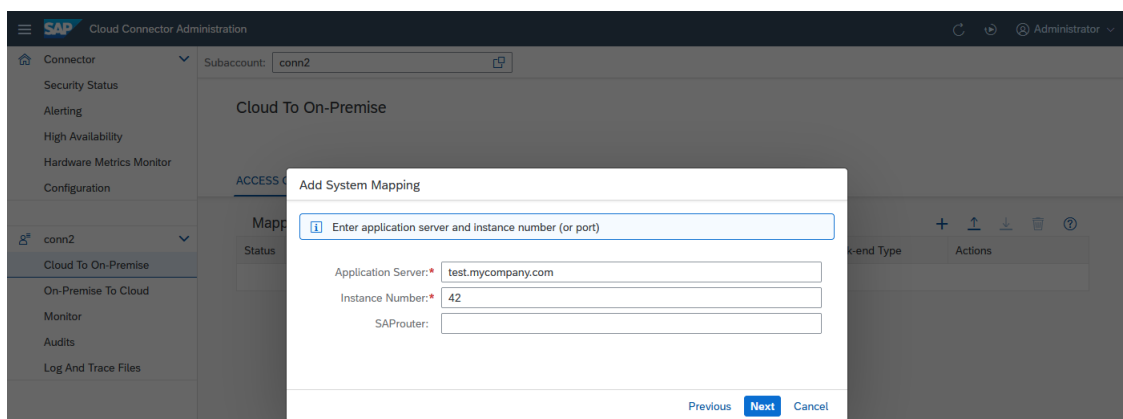
6. Choose **Next**.
7. Choose whether you want to configure a load balancing logon or connect to a specific application server.



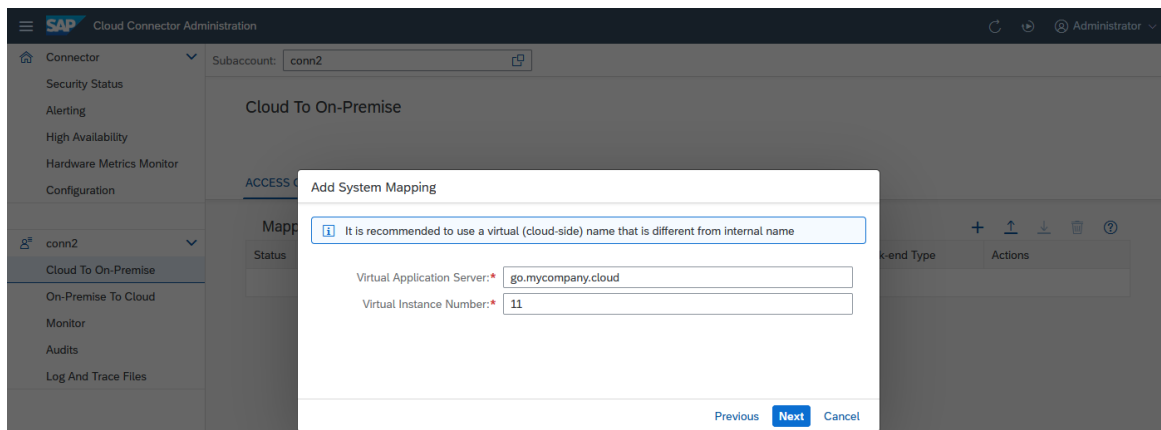
8. Specify the parameters of the backend system. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. If this is only possible using a valid [SAProuter](#), specify the router in the respective field. The Cloud Connector will try to establish a connection to this system, so the address has to be real.
 - When using a load-balancing configuration, the [Message Server](#) specifies the message server of the ABAP system. The system ID is a three-char identifier that is also found in the SAP Logon configuration. Alternatively, it's possible to directly specify the message server port in the [System ID](#) field.



- When using direct logon, the [Application Server](#) specifies one application server of the ABAP system. The instance number is a two-digit number that is also found in the SAP Logon configuration. Alternatively, it's possible to directly specify the gateway port in the [Instance Number](#) field.



9. Optional: You can virtualize the system information in case you like to hide your internal host names from the cloud. The virtual information can be a fake name which does not need to exist. The fields will be pre-populated with the values of the configuration provided in [Message Server](#) and [System ID](#), or [Application Server](#) and [Instance Number](#).

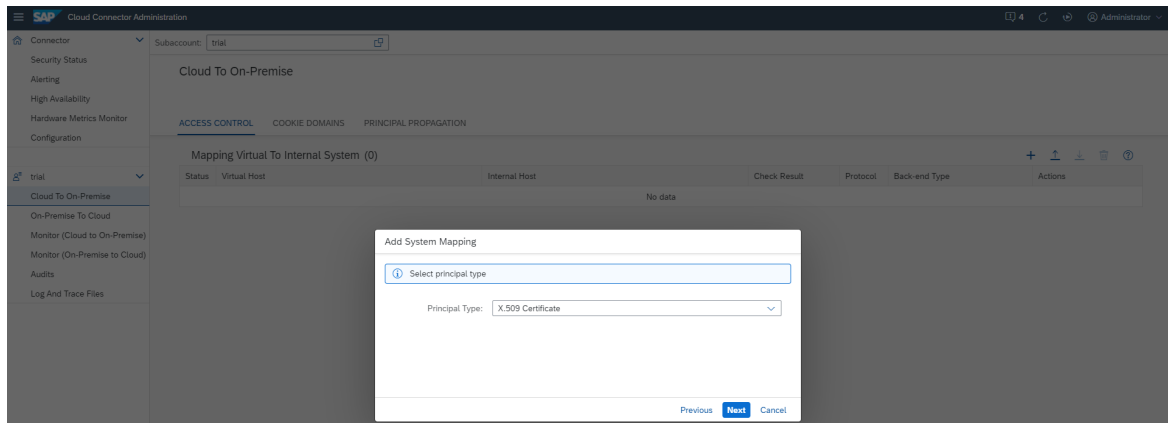


- [Virtual Message Server](#) - specifies the host name exactly as specified as the `jco.client.mshost` property in the RFC destination configuration in the cloud. The [Virtual System ID](#) allows you to distinguish between different entry points of your backend system that have different sets of access control settings. The value needs to be the same like for the `jco.client.r3name` property in the RFC destination configuration in the cloud.
 - [Virtual Application Server](#) - it specifies the host name exactly as specified as the `jco.client.ashost` property in the RFC destination configuration in the cloud. The [Virtual Instance Number](#) allows you to distinguish between different entry points of your backend system that have different sets of access control settings. The value needs to be the same like for the `jco.client.sysnr` property in the RFC destination configuration in the cloud.
10. This step is only relevant if you have chosen **RFC SNC**. The `<Principal Type>` field defines what kind of principal is used when configuring a destination on the cloud side using this system mapping with authentication type `Principal Propagation`. No matter what you choose, make sure that the general configuration for the `<Principal Type>` has been done to make it work correctly. For destinations using different authentication types, this setting is ignored. In case you choose **None** as `<Principal Type>`, it is not possible to apply principal propagation to this system.

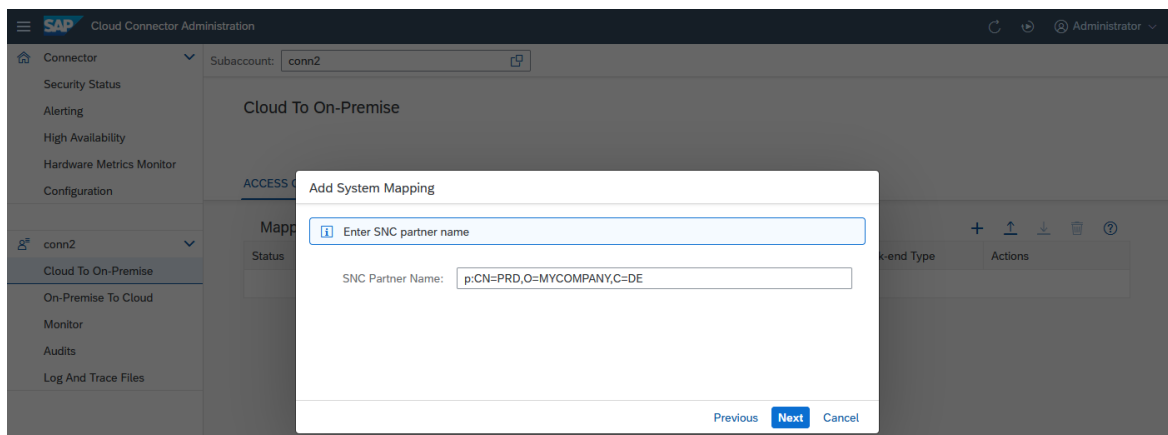
Note

If you use an RFC connection, you cannot choose between different principal types. Only the **x.509** certificate is supported. You need an SNC-enabled backend connection to use it.

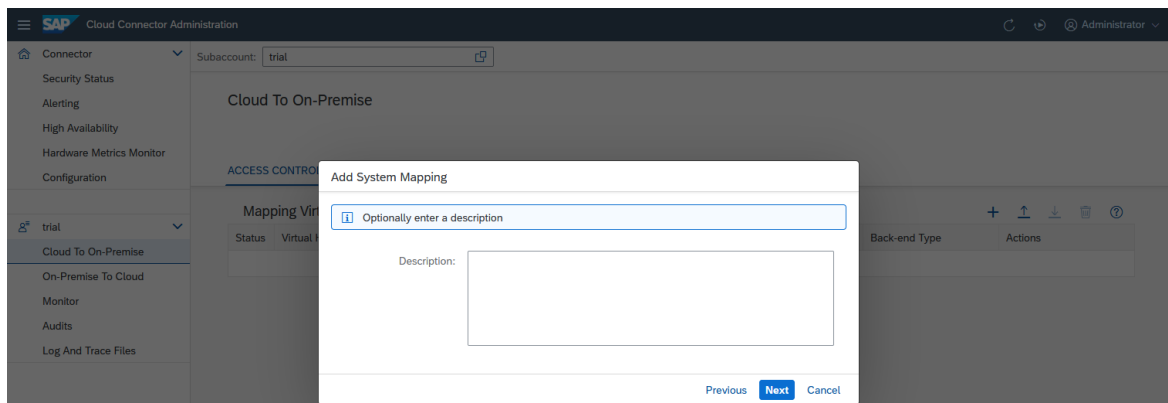
For more information on principal propagation, see [Configuring Principal Propagation \[page 304\]](#).



11. **SNC Partner Name:** This step will only come up if you have chosen **RFC SNC**. The SNC partner name needs to contain the correct SNC identification of the target system.



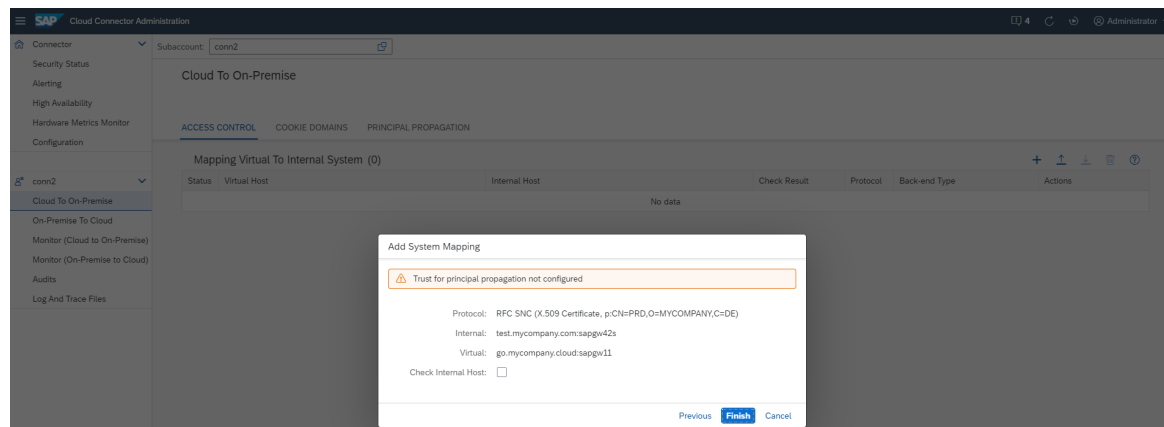
12. **Mapping Virtual to Internal System:** You can enter an optional description at this stage. The respective description will be shown as a rich tooltip when the mouse hovers over the entries of the virtual host column (table).



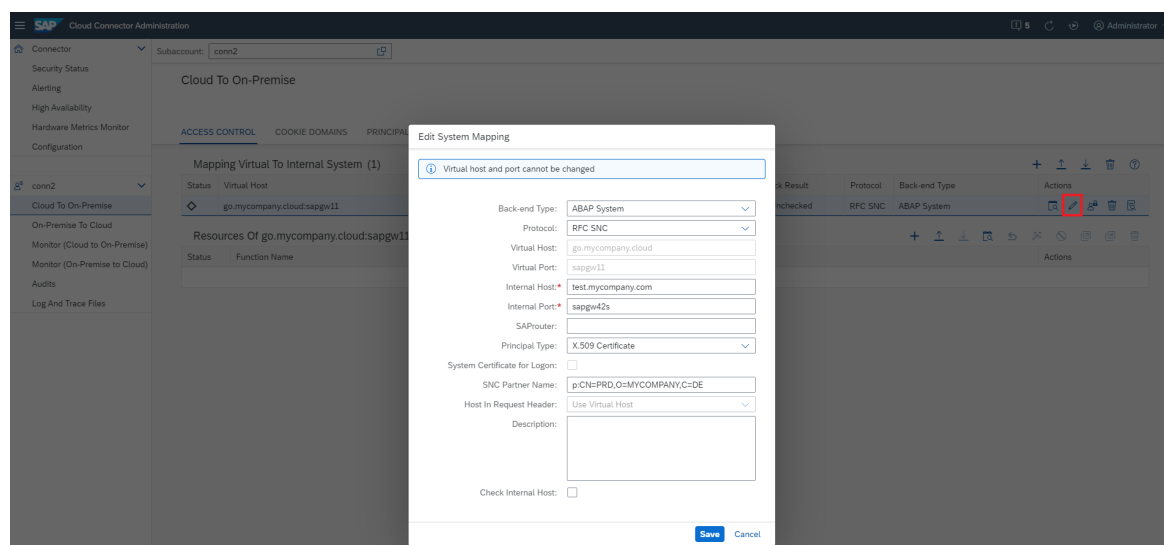
13. The summary shows information about the system to be stored. When saving the system mapping, you can trigger a ping from the Cloud Connector to the internal host, using the **Check Internal Host** checkbox. This allows you to make sure the Cloud Connector can indeed access the internal system, and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

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

You can execute such a check at any time later for all selected systems in the [Access Control](#) overview.



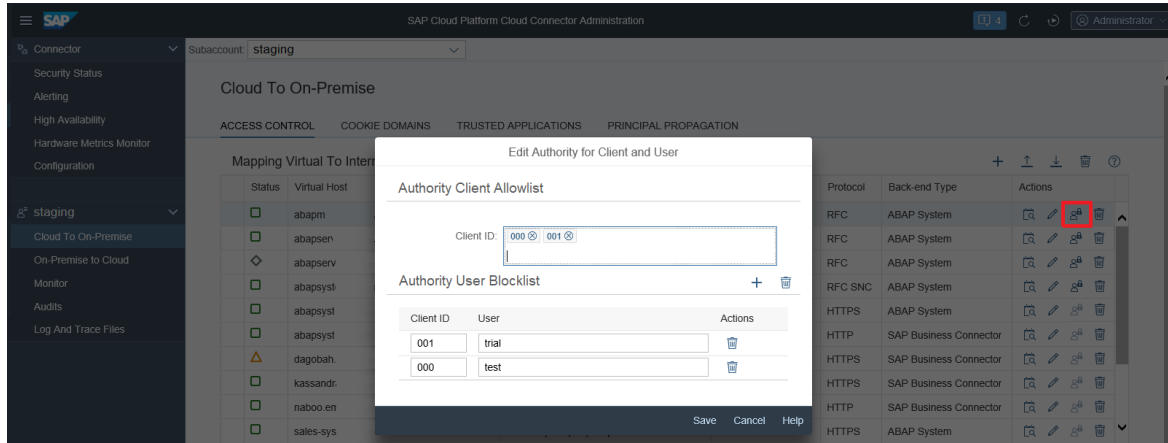
14. Optional: You can later edit a system mapping (choose [Edit](#)) to make the Cloud Connector route the requests for **sales-system.cloud:sapgw42** to a different backend system. This can be useful if the system is currently down and there is a back-up system that can serve these requests in the meantime. However, you cannot edit the virtual name of this system mapping. If you want to use a different fictional host name in your cloud application, you must delete the mapping and create a new one. Here, you can also change the [Principal Type](#) to **None** in case you don't want to allow principal propagation to a certain system.



15. Optional. You can later edit a system mapping to add more protection to your system when using RFC via the Cloud Connector, by restricting the mapping to specified clients and users: in column [Actions](#), choose the button [Maintain Authority Lists \(only RFC\)](#) to open an allowlist/blocklist dialog. In section [Authority Client Allowlist](#), enter all clients of the corresponding system in the field `<Client ID>` that you want to allow to use the Cloud Connector connection. In section [Authority User Blocklist](#), press the button [Add a user authority](#) (+) to enter all users you want to exclude from this connection. Each user must be assigned to a specified client. When you are done, press [Save](#).

Note

This function applies for RFC/RFC SNC only.

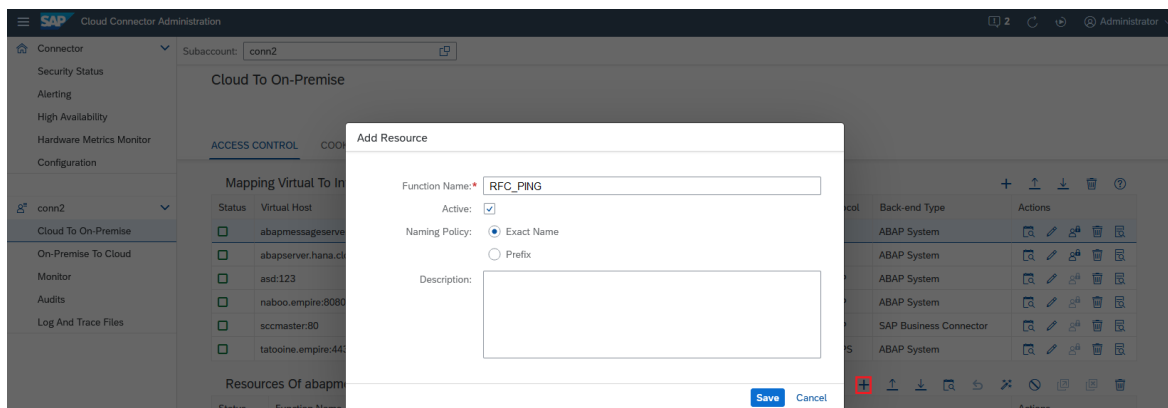


Back to [Tasks \[page 350\]](#)

Limit the Accessible Resources for RFC

In addition to allowing access to a particular host and port, you also must specify which function modules ([Resources](#)) are allowed to be invoked on that host. You can enter an optional description at this stage. The Cloud Connector uses very strict allowlists for its access control. Besides internally used infrastructure function modules, only function modules for which you explicitly granted access are allowed.

1. To define the permitted function modules for a particular backend system, choose the row corresponding to that backend system and press [Add](#) in section [Resources Accessible On...](#) below. A dialog appears, prompting you to enter the specific function module name whose invoking you want to allow.



2. The Cloud Connector checks that the function module name of an incoming request is exactly as specified in the configuration. If it is not, the request is denied.
3. If you select the [Prefix](#) option, the Cloud Connector allows all incoming requests, for which the function module name begins with the specified string.
4. The [Active](#) checkbox allows you to specify whether that resource should be initially enabled or disabled.

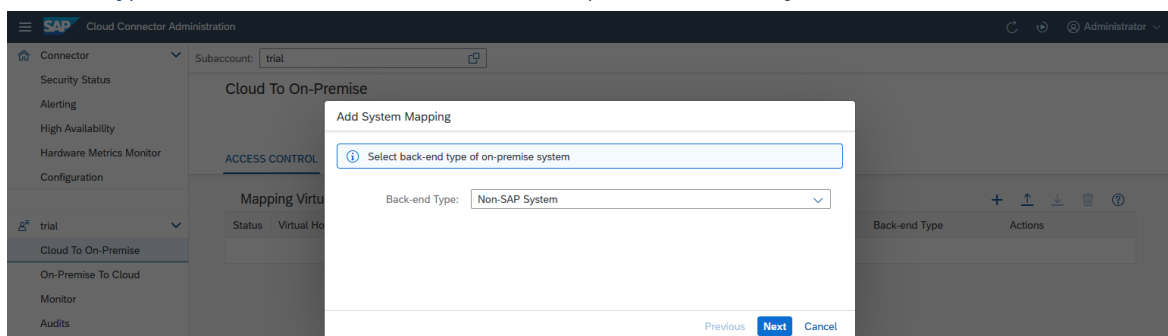
Back to [Tasks \[page 350\]](#)

1.5.2.4.3 Configure Access Control (LDAP)

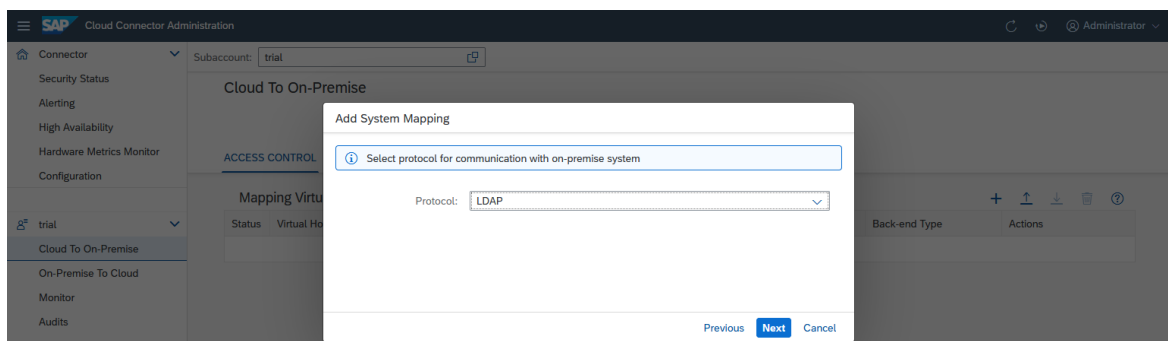
Add a specified system mapping to the Cloud Connector if you want to use an on-premise LDAP server or user authentication in your cloud application.

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

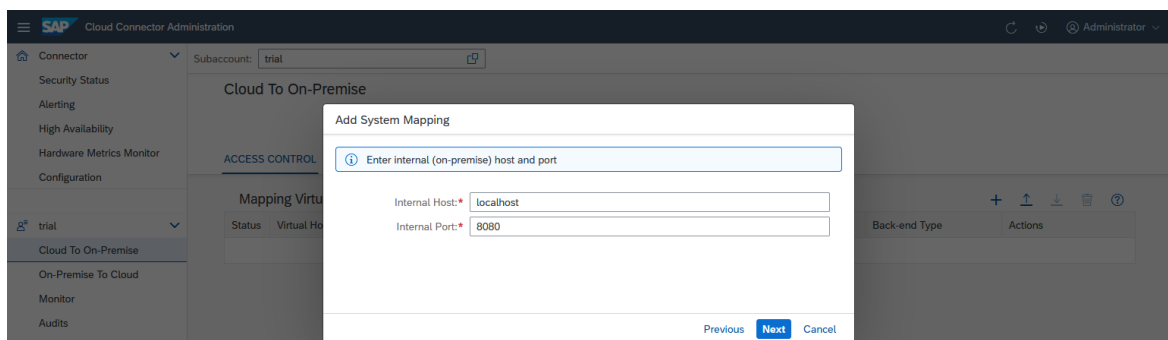
1. Choose *Cloud To On-Premise* from your *Subaccount* menu.
2. Choose *Add (+)*. A wizard opens and asks for the required values.
3. *Backend Type*: Select **Non-SAP System** from the drop down list. When you are done, choose *Next*.



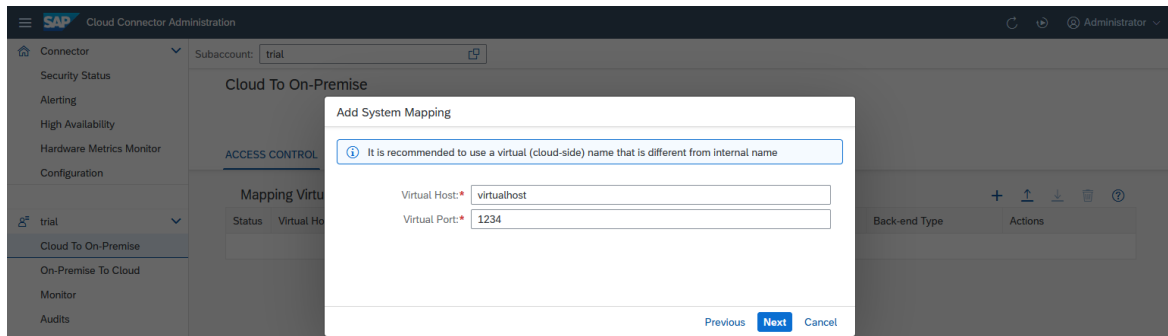
4. *Protocol*: Select **LDAP** or **LDAPS** for the connection to the backend system. When you are done, choose *Next*.



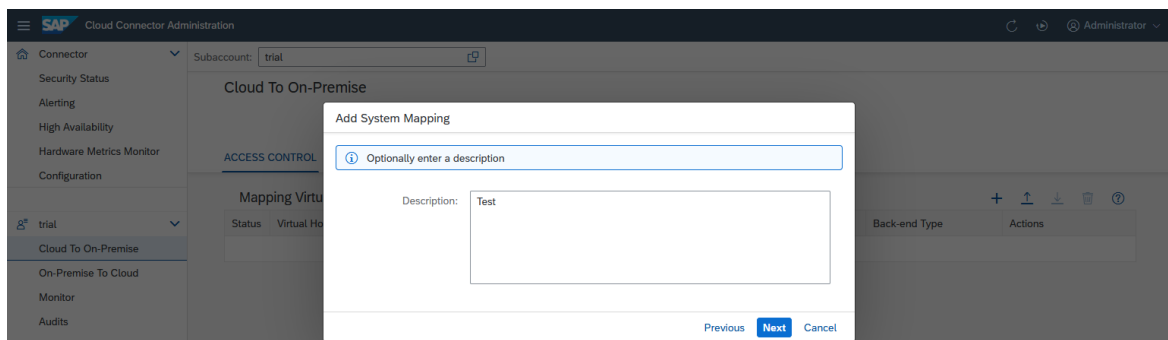
5. *Internal Host* and *Internal Port*: specify the host and port under which the target system can be reached within the intranet. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. The Cloud Connector will try to forward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

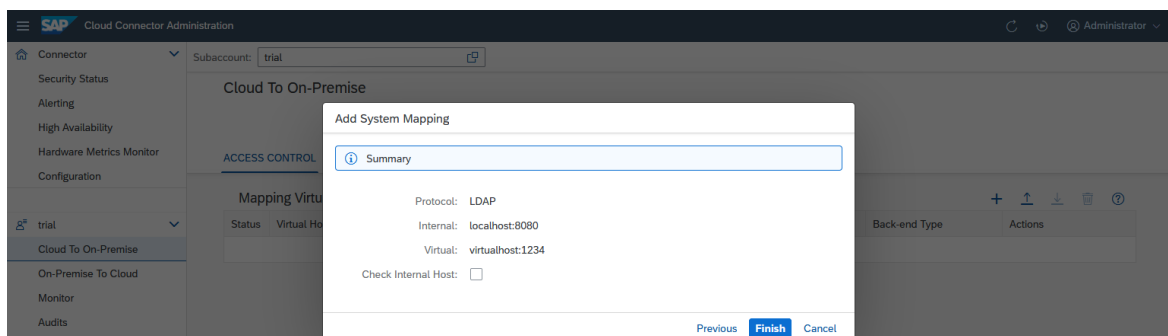


7. You can enter an optional description at this stage. The respective description will be shown as a tooltip when you press the button *Show Details* in column *Actions* of the *Mapping Virtual To Internal System* overview.

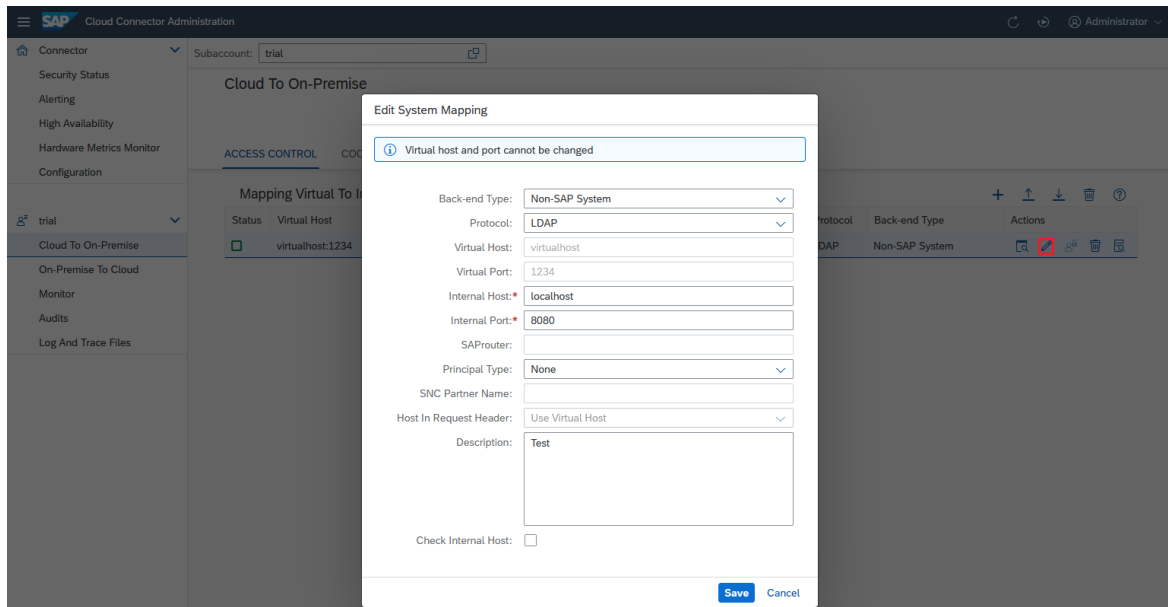


8. The summary shows information about the system to be stored. When saving the host mapping, you can trigger a ping from the Cloud Connector to the internal host, using the *Check Internal Host* check box. This allows you to make sure the Cloud Connector can indeed access the internal system. Also,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

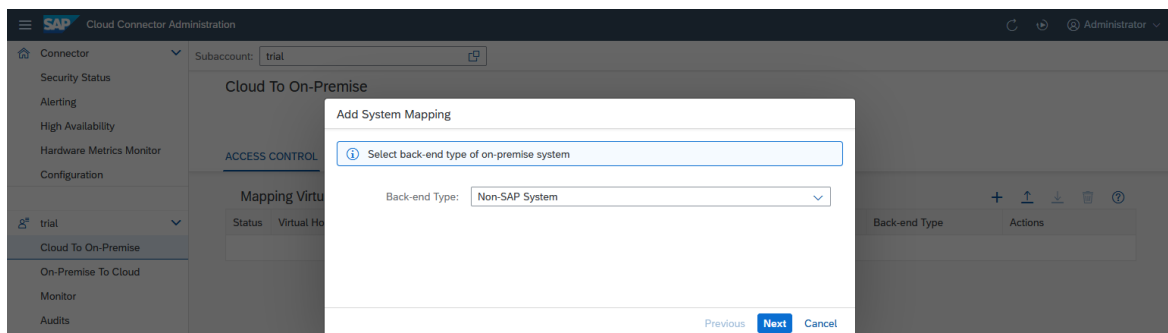


1.5.2.4.4 Configure Access Control (TCP)

Add a specified system mapping to the Cloud Connector if you want to use the TCP protocol for communication with a backend system.

To allow your cloud applications to access a certain backend system on the intranet via TCP, insert a new entry in the Cloud Connector access control management.

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**.

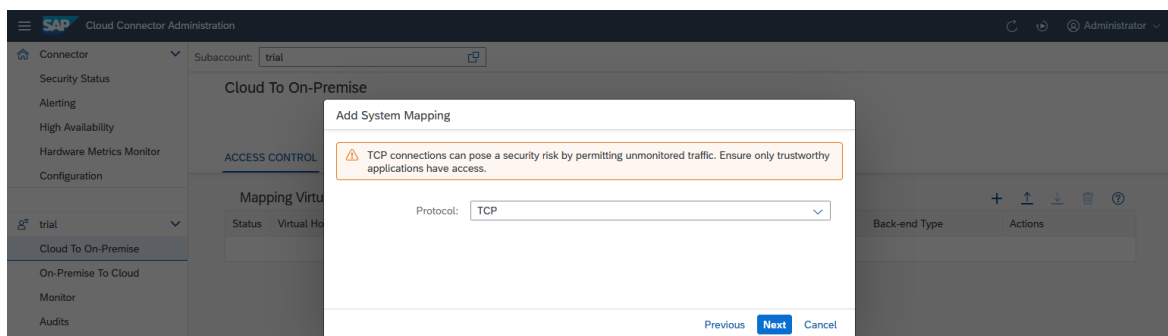


4. **Protocol**: Select **TCP** or **TCP SSL** for the connection to the backend system. When choosing TCP, you can perform an end-to-end TLS handshake from the cloud client to the backend. If the cloud-side client is using plain communication, but you still need to encrypt the hop between Cloud Connector and the backend, choose **TCP SSL**. When you are done, choose **Next**.

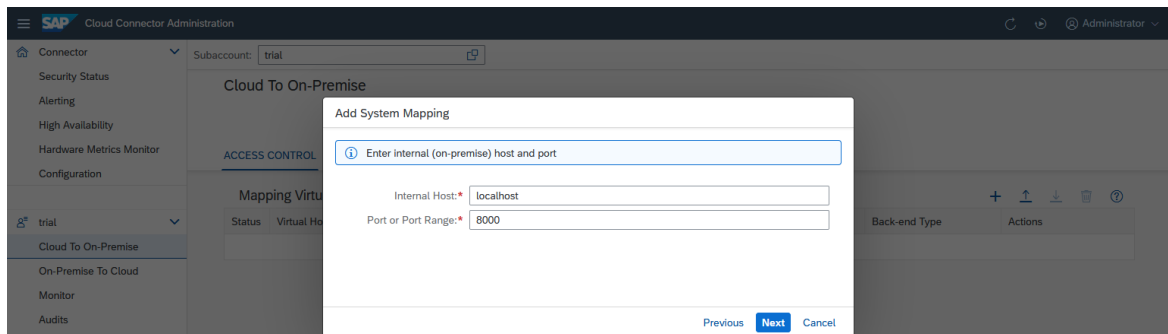
Note

When selecting TCP as protocol, the following warning message is displayed: TCP connections can pose a security risk by permitting unmonitored traffic. Ensure only trustworthy applications have access. The reason is that using plain TCP, the Cloud Connector cannot see or log any detail information about the calls. Therefore, in contrast to HTTP or RFC (both running on top of TCP), the Cloud Connector cannot check the validity of a request. To minimize this risk, make sure you

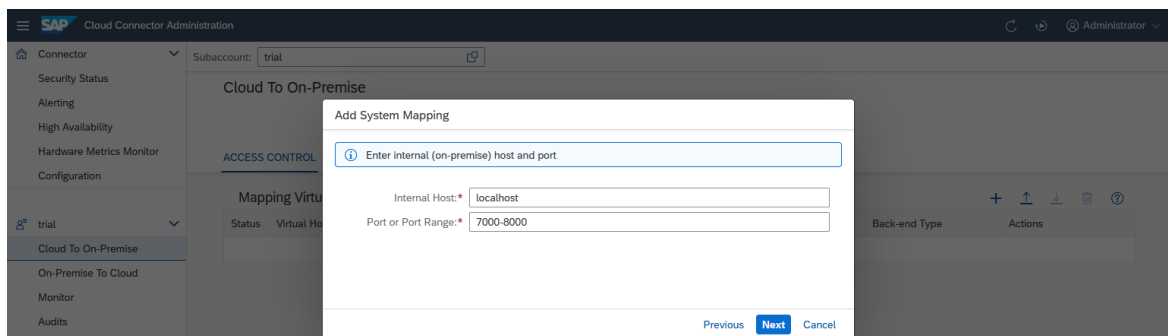
- deploy only trusted applications on SAP BTP.
- configure an application allowlist in the Cloud Connector, see [Set Up Trust \[page 304\]](#).
- take the recommended security measures for your SAP BTP (sub)account. See section [Security](#) in the SAP BTP documentation.



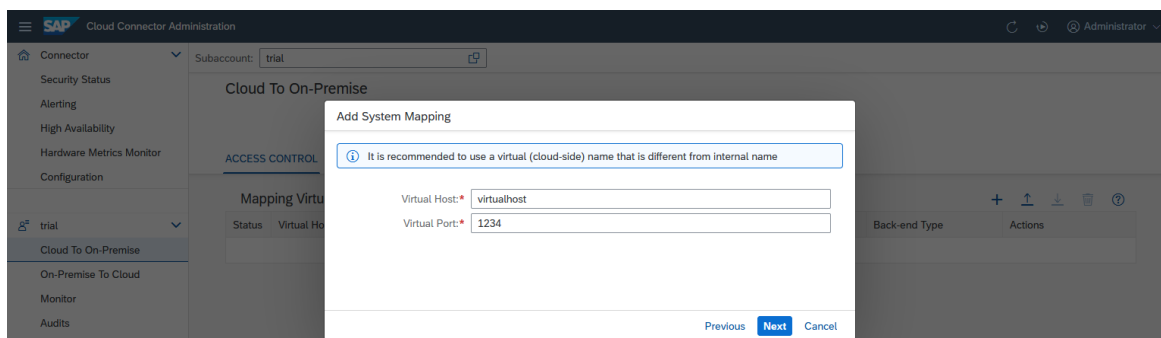
5. **Internal Host** and **Port or Port Range**: specify the host and port under which the target system can be reached within the intranet. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. The Cloud Connector will try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



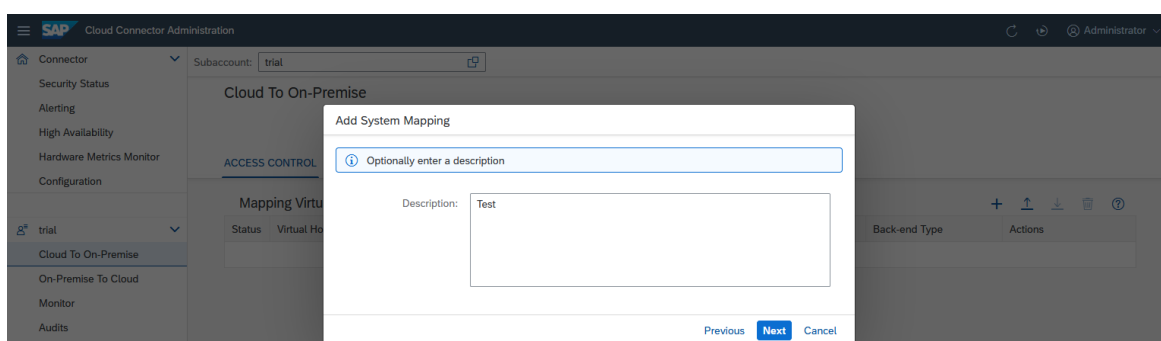
For TCP and TCP SSL, you can also specify a port range through its lower and upper limit, separated by a hyphen.



- 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*.



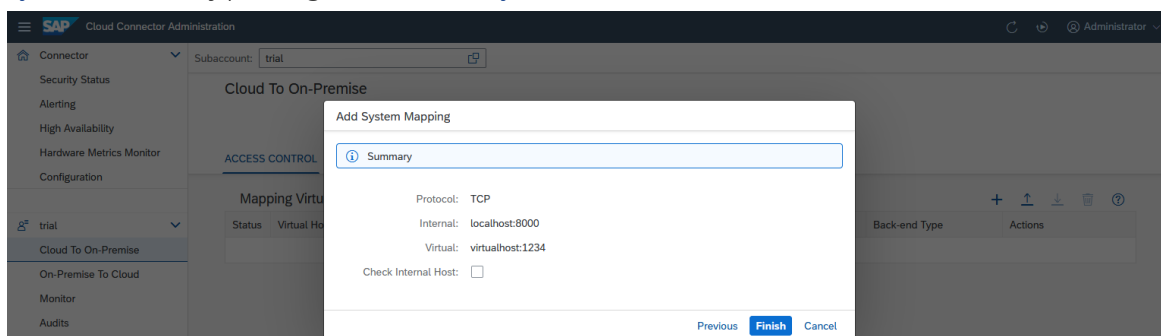
- You can enter an optional description at this stage. The respective description will be shown as a tooltip when you press the button *Show Details* in column *Actions* of the *Mapping Virtual To Internal System* overview.



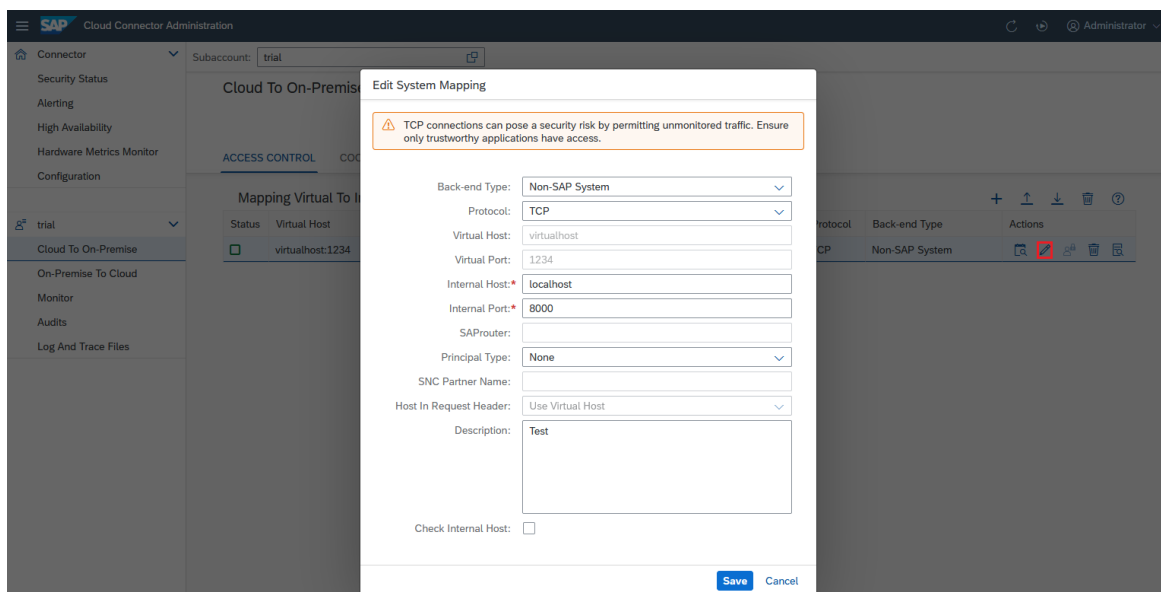
- The summary shows information about the system to be stored. When saving the host mapping, you can trigger a ping from the Cloud Connector to the internal host, using the *Check Internal Host* checkbox. This allows you to make sure the Cloud Connector can access the internal system. Also, you can catch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



- Optional: You can later edit the system mapping (by choosing [Edit](#)) to make the Cloud Connector route the requests to a different backend system. This can be useful if the system is currently down and there is a backup system that can serve these requests in the meantime. However, you cannot edit the virtual name nor port of this system mapping. If you want to use a different fictional host name in your cloud application, you must delete the mapping and create a new one. The same goes for port ranges. If a port range needs to be changed, you must delete the mapping and create it again with the desired port range.



1.5.2.4.5 Configure Accessible Resources

Configure backend systems and resources in the Cloud Connector, to make them available for a cloud application.

Tasks

[Map Systems and Limit Access \[page 362\]](#)

[Use Scenarios for Resources \[page 364\]](#)

Map Systems and Limit Access

Initially, after installing a new Cloud Connector, no network systems or resources are exposed to the cloud. You must configure each system and resource used by applications of the connected cloud subaccount. To do this, choose [Cloud To On Premise](#) from your subaccount menu and go to tab [Access Control](#):

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar contains navigation options: Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, trial, Cloud To On-Premise, On-Premise To Cloud, Monitor, Audits, and Log And Trace Files. The main area is titled 'Cloud To On-Premise' and includes tabs for ACCESS CONTROL, COOKIE DOMAINS, APPLICATIONS, and PRINCIPAL PROPAGATION. The 'ACCESS CONTROL' tab is active, displaying a table titled 'Mapping Virtual To Internal System (7)'. This table lists virtual hosts, internal hosts, check results, protocols, and back-end types. Below this, a red-bordered box highlights a table titled 'Resources Of abapmessagesserver.hana.cloud:sapmsFCB (4)', which lists function names and naming policies.

Status	Virtual Host	Internal Host	Check Result	Protocol	Back-end Type	Actions
<input type="checkbox"/>	abapmessagesserver.hana.cloud:sapmsFCB	xxxxx.wdf.sap.corp:sapmsV9U	Unchecked	RFC	ABAP System	[Icons]
<input type="checkbox"/>	abapserver.hana.cloud:sapgw42	xxxxx.wdf.sap.corp:sapgw51	Unchecked	RFC	ABAP System	[Icons]
<input type="checkbox"/>	asd:123	asd:123	Unchecked	HTTP	ABAP System	[Icons]
<input type="checkbox"/>	ldciv9u.wdf.sap.corp:sapgw51	xxxxx.wdf.sap.corp:sapgw51	Unchecked	RFC	ABAP System	[Icons]
<input type="checkbox"/>	naboo.empire:8080	xxxxx.dhcp.wdf.sap.corp:5555	Unchecked	HTTP	ABAP System	[Icons]
<input type="checkbox"/>	sccmaster:80	xxxxx.wdf.sap.corp:443	Unchecked	HTTP	SAP Business Connector	[Icons]
<input type="checkbox"/>	tatooline.empire:443	xxxxx.dhcp.wdf.sap.corp:4443	Unchecked	HTTP	ABAP System	[Icons]

Status	Function Name	Naming Policy	Actions
<input type="checkbox"/>	JTEST_ADD_INT	Exact Name	[Icons]
<input type="checkbox"/>	RFC_RAISE	Prefix	[Icons]
<input type="checkbox"/>	SBC_STRING	Exact Name	[Icons]
<input type="checkbox"/>	SFTC_CONNECTION	Exact Name	[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 \(HTTP\) \[page 342\]](#).
- For information on configuring RFC resources, see: [Configure Access Control \(RFC\) \[page 350\]](#).

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

 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:*	<input type="text" value="xxxxx.wdf.sap.corp"/>
Internal Port:*	<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 \[page 362\]](#)

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 \[page 365\]](#)

[Import a Scenario \[page 365\]](#)

[Remove a Scenario \[page 366\]](#)

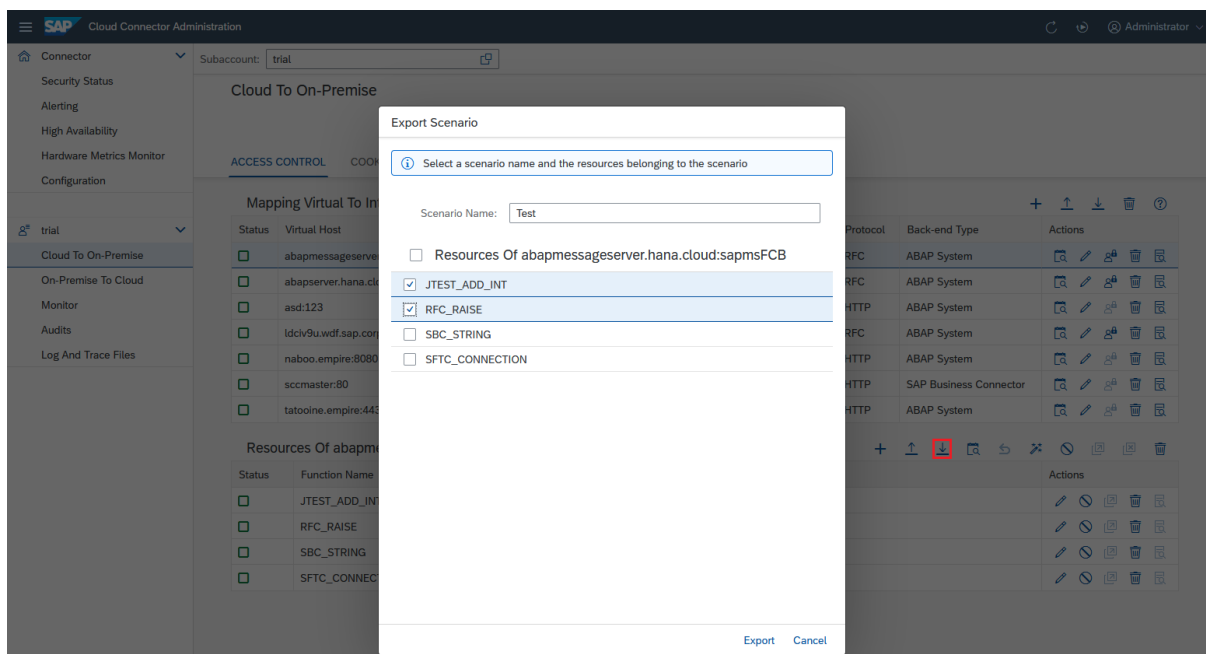
Define and Export a Scenario

If you, as application owner, have implemented and tested a scenario, and configured a Cloud Connector accordingly, you can define the scenario as follows:

1. Choose the [Export Scenario](#) button.
2. In the dialog, select the resources that belong to the scenario.
3. In the field **<Scenario Name>**, choose a descriptive name, under which the set of resources can be identified in the consuming Cloud Connector.
4. Choose [Export](#) to store the scenario.

Note

For applications provided by SAP, default scenario definitions may be available. To verify this, check the corresponding application documentation.



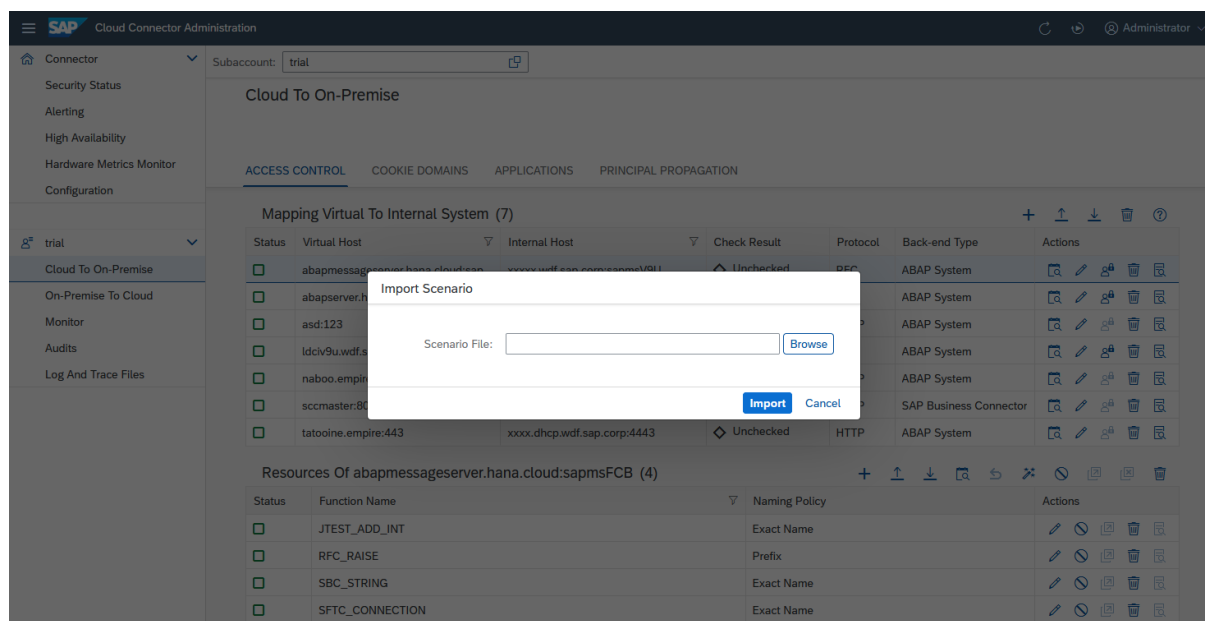
Back to [Use Scenarios for Resources \[page 364\]](#)

Back to [Tasks \[page 362\]](#)

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.



All resources belonging to a scenario get an additional scenario icon in their status. When hovering over it, the assigned scenario(s) of this resource are listed.

Back to [Use Scenarios for Resources \[page 364\]](#)

Back to [Tasks \[page 362\]](#)

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.

Cloud Connector Administration

Administrator

Connector

Security Status

Alerting

High Availability

Hardware Metrics Monitor

Configuration

trial

Cloud To On-Premise

On-Premise To Cloud

Monitor

Audits

Log And Trace Files

Subaccount: trial

Cloud To On-Premise

ACCESS CONTROL

COOKIE DOMAINS

APPLICATIONS

PRINCIPAL PROPAGATION

Mapping Virtual To Internal System (6)

Status	Virtual Host	Internal Host	Check Result	Protocol	Back-end Type	Actions
<input type="checkbox"/>	abapmessagesserver.hana.cloud:sapms...	xxxxx.wdf.sap.corp:sapmsV9U	Unchecked	RFC	ABAP System	
<input type="checkbox"/>	abapserver.hana.cloud:sapgw42	xxxxx.wdf.sap.corp:sapgw51	Unchecked	RFC	ABAP System	
<input type="checkbox"/>	asd:123	asd:123	Unchecked	HTTP	ABAP System	
<input type="checkbox"/>	naboo.empire:8080	xxxxx.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	

Resources Of asd:123 (3)

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

Back to [Use Scenarios for Resources \[page 364\]](#)

Back to [Tasks \[page 362\]](#)

1.5.2.5 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 \[page 278\]](#)
- [Managing Subaccounts \[page 291\]](#)
- [Configure Access Control \[page 340\]](#)

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 \[page 368\]](#)
- [High Availability Settings \[page 370\]](#)
- [Proxy Settings \[page 381\]](#)
- [Authentication and UI Settings \[page 384\]](#)
- [Certificate Management for Backend Communication \[page 396\]](#)
- [Solution Management Configuration \[page 409\]](#)
- [Backup \[page 412\]](#)
- [Subaccount \[page 413\]](#)
- [System Mappings \[page 425\]](#)
- [System Mapping Resources \[page 430\]](#)
- [Domain Mappings \[page 442\]](#)
- [Subaccount Service Channels \[page 445\]](#)
- [Access Control Entities \[page 465\]](#)

Related Information

[Examples \[page 472\]](#)

1.5.2.5.1 Common Properties

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

Get Common Properties

URI	<code>/api/v1/configuration/connector</code>
Method	<code>GET</code>
Request	
Response	<code>{ha, description}</code>
Errors	

- **Response Properties:**

- `ha`: Cloud Connector instance assigned to a high-availability role. Its value is either the string *master* or *shadow*.
- `description`: Description of the Cloud Connector.

Get Version

Note

Available as of version 2.14.0.

URI	<code>/api/v1/connector/version</code>
Method	<code>GET</code>
Request	
Response	<code>{version}</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

- **Response Properties:**

`version`: A string; the Cloud Connector version.

Set Description (Master Only)

URI	<code>/api/v1/configuration/connector</code>
Method	<code>PUT</code>
Request	<code>{description}</code>

Response	<code>{ha, description}</code>
Errors	INVALID_REQUEST
Roles	Administrator

- **Request Properties:**
`description`: A string; use an empty string to remove the description.
- **Response Properties:**
 - `ha`: Cloud Connector instance assigned to a high-availability role. Its value is either the string *master* or *shadow*.
 - `description`: Description of the Cloud Connector.
- **Errors:**
INVALID_REQUEST ((400): The value of `description` is not a JSON string.

Note

`null` is not a JSON string.

1.5.2.5.2 High Availability Settings

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

When installing a Cloud Connector instance, you usually define its high availability role (master or shadow instance) during initial configuration, see [Change your Password and Choose Installation Type \[page 280\]](#).

If the high availability role was not defined before, you can set the master or shadow role via this API.

If a shadow instance is connected to the master, this API also lets you switch the roles: the master instance requests the shadow instance to take over the master role, and then takes the shadow role itself.

Note

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

Get High Availability Settings

URI	<code>/api/v1/configuration/connector/haRole</code>
-----	---

Method	<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 https://localhost:8443/api/v1/configuration/connector/haRole
```

Related Information

[Master Instance Configuration \[page 372\]](#)

[Shadow Instance Configuration \[page 376\]](#)

1.5.2.5.2.1 Master Instance Configuration

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

Note

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

Restriction

These APIs are only permitted on a Cloud Connector master instance. The shadow instance rejects the requests with error code *400 – Invalid Request*.

Get Configuration

URI	/api/v1/configuration/connector/ha/master/config
Method	GET
Request	
Response	{haEnabled, haPort, allowedShadowHost}
Errors	

Response Properties:

- `haEnabled`: Boolean value that indicates whether or not a shadow system is allowed to connect.
- `haPort`: Port on which the shadow instance can connect (restart required after change).
- `allowedShadowHost`: Name of the shadow host (a string) that is allowed to connect; an empty string signifies that any host is allowed to connect as shadow.

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/configuration/connector/ha/master/config
```

Set Configuration

URI	/api/v1/configuration/connector/ha/master/config
Method	PUT
Request	{haEnabled, haPort, allowedShadowHost}
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Response Properties:

- `haEnabled`: Boolean value that indicates whether or not a shadow system is allowed to connect.
- `haPort`: Port on which the shadow instance can connect (restart required after change).
- `allowedShadowHost`: Name of the shadow host (a string) that is allowed to connect. An empty string means that any host is allowed to connect as shadow.

Errors:

- `INVALID_REQUEST` (400): if the name of the shadow host is not a valid host name

Example

```
curl -i -k -u <user>:<password> -X PUT -H 'Content-Type: application/json'
-d '{"haEnabled":true}' https://<host>:<port>/api/v1/configuration/connector/ha/
master/config
```

Get State

URI	/api/v1/configuration/connector/ha/ master/state
Method	GET
Request	
Response	{state, shadowHost}
Errors	
Roles	Administrator, Display, Support

Response Properties:

- **state**: One of the following strings: ALONE, BINDING, CONNECTED or BROKEN.
- **shadowHost**: Connected shadow host (a string)

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/
configuration/connector/ha/master/state
```

Set State

URI	/api/v1/configuration/connector/ha/ master/state
Method	POST

Request	
Response	{op}
Errors	ILLEGAL_STATE, INVALID_REQUEST
Roles	Administrator

Request Properties:

The value of property `op` is one of the following strings:

- `SWITCH`: Switch roles with shadow
- `FORCE_SWITCH`: Take over the shadow role, even if shadow instance does not respond.

Errors:

- `INVALID_REQUEST` (400): Value of property `op` is neither `SWITCH` nor `FORCE_SWITCH`.
- `ILLEGAL_STATE` (409): Master system is in a state that does not permit the requested operation.

Example

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

Reset

A successful call to this API restores default values for all settings related to high availability on the master side.

⚠ Caution

Do not perform this call if the shadow is connected to a master.

URI	/api/v1/configuration/connector/ha/master/state
Method	DELETE
Request	
Response	204 on success

Errors	ILLEGAL_STATE
Roles	Administrator

Errors:

- `ILLEGAL_STATE` (409): A shadow instance is connected to the master.

Example

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

1.5.2.5.2.2 Shadow Instance Configuration

Read and edit the configuration settings for a Cloud Connector shadow instance via API (available as of Cloud Connector version 2.12.0, or, where mentioned, as of version 2.13.0).

Note

The APIs below are only permitted on a Cloud Connector shadow instance. The master instance will reject the requests with error code 403 – `FORBIDDEN_REQUEST`.

Get Configuration

URI	<code>/api/v1/configuration/connector/ha/shadow/config</code>
Method	<code>GET</code>
Request	
Response	<pre>{masterHost, masterPort, ownHost, haPort, checkIntervalInSeconds, takeoverDelayInSeconds, connectTimeoutInMillis, requestTimeoutInMillis}</pre>
Errors	

Response Properties:

- `masterHost`: Host name of the master instance (string)
- `masterPort`: Port of the master instance (string or number)
- `ownHost`: Host name of the shadow instance (string)
- `haPort`: Own port for HA communication (restart required after change).
- `checkIntervalInSeconds`: Time between two health checks against the master instance (number)
- `takeoverDelayInSeconds`: The time a master instance may stay unreachable before the shadow instance takes over (number)
- `connectTimeoutInMillis`: Timeout for connection attempts between shadow and master instance (number)
- `requestTimeoutInMillis`: Timeout for requests between shadow and master instance (number)

Note

This API may take some time to fetch the own hosts from the environment.

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/configuration/connector/ha/shadow/config
```

Set Configuration

URI	/api/v1/configuration/connector/ha/shadow/config
Method	PUT
Request	<pre>{masterPort, masterHost, ownHost, haPort, checkIntervalInSeconds, takeoverDelayInSeconds, connectTimeoutInMillis, requestTimeoutInMillis}</pre>

Response

```
{masterPort, masterHost, ownHost,
haPort, checkIntervalInSeconds,
takeoverDelayInSeconds,
connectTimeoutInMillis,
requestTimeoutInMillis}
```

Errors

Roles

Administrator

Request Properties:

- `masterHost`: Host name of the master instance (string)
- `masterPort`: Port of the master instance (string or number)
- `ownHost`: Host name of the shadow instance (string)
- `haPort`: Own port for HA communication (restart required after change).
- `checkIntervalInSeconds`: Time between two health checks against the master instance (number)
- `takeoverDelayInSeconds`: The time a master instance may stay unreachable before the shadow instance takes over (number)
- `connectTimeoutInMillis`: Timeout for connection attempts between shadow and master instance (number)
- `requestTimeoutInMillis`: Timeout for requests between shadow and master instance (number)

Response Properties:

See *Request Properties*.

Example

```
curl -i -k -u <user>:<password> -X PUT https://<host>:<port>/api/v1/
configuration/connector/ha/shadow/config
-H 'Content-Type: application/json' -d '{"masterHost":"localhost",
"masterPort":"8443", "ownHost":"localhost",
"checkIntervalInSeconds":30, "takeoverDelayInSeconds":10,
"connectTimeoutInMillis":1000,
"requestTimeoutInMillis":12000}'
```

Get State

📘 Note

Available as of version 2.13.0.

URI	/api/v1/configuration/connector/ha/shadow/state
Method	GET
Request	
Response	<pre>{state, ownHosts, stateMessage, masterVersions}</pre>
Errors	
Roles	Administrator, Display, Support

Response Properties:

- **state**: Possible string values are: INITIAL, DISCONNECTED, DISCONNECTING, HANDSHAKE, INITSYNC, READY, or LOST.
- **ownHosts**: List of alternative host names for the shadow instance.
- **stateMessage**: Message providing details on the current state. This property may not always be present. Typically, this property is available if an error occurred (for example, a failed attempt to connect to the master instance).
- **masterVersions**: Overview of relevant component versions of the master system, including a flag (*property ok*) that indicates whether or not there are incompatibility issues because of differing master and shadow versions.

Note

This property is only available if the shadow instance is connected to the master instance, or if there has been a successful connection to the master system at some point in the past.

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/configuration/connector/ha/shadow/state
```

Change State

URI	/api/v1/configuration/connector/ha/shadow/state
Method	POST

Request	
Response	<code>{op, user, password}</code>
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.

Note

The logon credentials are used for initial logon to master instance only. If a shadow instance is disconnected from its master instance, it will reconnect to the (same) master instance using a certificate. Hence, user and password can be omitted when reconnecting.

Example

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

Reset

Note

Available as of version 2.13.0.

A successful call to this API deletes master host and port, and restores default values for all other settings related to a connection to the master.

⚠ Caution

Do not perform this call if the shadow is connected to a master.

URI	/api/v1/configuration/connector/ha/shadow/state
Method	DELETE
Request	
Response	
Errors	ILLEGAL_STATE
Roles	Administrator

Errors:

- ILLEGAL_STATE (409): the shadow is currently connected to a master.

Example

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

1.5.2.5.3 Proxy Settings

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

Get Proxy Settings

URI	/api/v1/configuration/connector/proxy
Method	GET
Request	

Response	<code>{host, port, user}</code>
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	<code>/api/v1/configuration/connector/proxy</code>
Method	<i>PUT</i>
Request	<code>{host, port, user, password}</code>
Response	
Errors	INVALID_REQUEST, FORBIDDEN_REQUEST
Roles	Administrator

Request Properties:

- `host`: the name of the proxy host (a string)
- `port`: the port of the proxy host (a string)
- `user`: the user name (a string)
- `password`: the password (a string - optional)

Errors:

- `INVALID_REQUEST` (400): invalid values were supplied, or mandatory values are missing.
- `FORBIDDEN_REQUEST` (403): the target of the call is a shadow instance.

The following example sets empty `user` and `password`.

Sample Code

```
curl -ik -u Administrator:<password> https://localhost:8443/api/v1/
configuration/connector/proxy -X PUT -H 'Content-Type: application/json' -d
'{"host":"proxy", "port":"8080"}'
```

This request removes the proxy configuration.

Sample Code

```
curl -ik -u Administrator:<password> https://localhost:8443/api/v1/
configuration/connector/proxy -X PUT -H 'Content-Type: application/json' -d
'{'
```

Remove Proxy Settings (Master Only)

URI	/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 -X DELETE
```

1.5.2.5.4 Authentication and UI Settings

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

Get Authentication Settings

URI	/api/v1/configuration/connector/authentication
Method	GET
Request	
Response	{type, configuration}
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `type`: The authentication type, which is one of the following strings: *basic* or *ldap*.
- `configuration`: The configuration of the active LDAP authentication. This property is only available if `type` is *ldap*. Its value is an object with properties that provide details on LDAP configuration.

Example

```
curl -i -k -H 'Accept:application/json'
-u Administrator:<password> -X GET https://<scchost>:8443/api/v1/configuration/
connector/authentication
```

Change Basic Authentication User

URI	/api/v1/configuration/connector/authentication/basic
Method	PUT

Request	<code>{password, user}</code>
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `password`: The current password (a string)
- `user`: The new user name (a string), overwriting the current user name.

Errors:

- `INVALID_REQUEST` (400): Current password is wrong.

Change Basic Authentication Password

URI	<code>/api/v1/configuration/connector/authentication/basic</code>
Method	<i>PUT</i>
Request	<code>{oldPassword, newPassword}</code>
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `oldPassword`: The current password, about to be changed (a string)
- `newPassword`: The new password (a string)

Errors:

- `INVALID_REQUEST` (400): Passwords are the same or current password is wrong.

Example

```
curl -i -k -H 'Content-Type:application/json' -d '{"oldPassword":"manage",
"newPassword":"test"}'
-u Administrator:manage -X PUT https://localhost:8443/api/v1/configuration/
connector/authentication/basic
```

Change LDAP Authentication

⚠ Caution

The Cloud Connector will restart if the request was successful. There is no test that confirms login will work afterwards. If you run into problems you can revert to basic authentication by executing the script `useFileUserStore` located in the root directory of your Cloud Connector installation.

URI	/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).

ⓘ Note

In both error cases nothing is stored, and LDAP authentication is disabled.

Example

```
curl -i -k -H 'Content-Type: application/json' -u Administrator:<password>
-X PUT https://localhost:8443/api/v1/configuration/connector/authentication/
ldap -d '{"enable":true, "configuration":{"hosts":[{"host":"ldaphost",
"port":"10389", "isSecure":false}], "config":{"roleBase=\"ou=groups,dc=scc\"
roleName=\"cn\" roleSearch=\"(uniqueMember={0})\"
userBase=\"ou=users,dc=scc\" userSearch=\"(uid={0})\"\", \"user\":\"ldapadmin\",
\"password\":\"<ldapadminpassword>\"}}}'
```

Get Description for UI Certificate

This API returns a textual description for the system certificate.

ⓘ Note

Available as of version 2.13.0.

URI	<code>/api/v1/configuration/connector/ui/uiCertificate</code>
Method	<code>GET</code>
Response	<code>{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}</code>
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `subjectDN`: The subject distinguished name (a string)
- `issuer`: The issuer (a string)
- `notBeforeTimeStamp`: Timestamp of the beginning of the validity period (a UTC long number)
- `notAfterTimeStamp`: Timestamp of the end of the validity period (a UTC long number)

- `subjectAltNames`: Subject alternative names, as an array of objects with properties `type` and `value`. The value of property `type` is one of the following strings: IP, DNS, URI, or RFC822. The value of property `value` is the associated value (a string).

Note

`subjectAltNames` is not present if there are no subject alternative names.

Example

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

Create a Self-Signed UI Certificate

Note

Available as of version 2.13.0.

URI	/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.

Note

This property is available as of version 2.15.0.

- `subjectDN`: The subject distinguished name (a string)


- `subjectAltNames`: Subject alternative names, as an array of objects with properties `type` and `value`. The value of property `type` is one of the following strings: IP, DNS, URI, or RFC822. The value of property `value` is the associated value (a string). This property is optional.

The UI certificate created this way has a validity of 1 year.

Example

```
curl -k -u Administrator:<password> -X POST -H "Content-Type: application/json"
--data '{"type":"selfsigned", "subjectDN":"CN=me"}'
https://<host>:<port>/api/v1/configuration/connector/ui/uiCertificate
```

Create a Certificate Signing Request for a UI Certificate



Note

Available as of version 2.13.0.

URI	/api/v1/configuration/connector/ui/uiCertificate
Method	POST
Request	{type, keySize, subjectDN, subjectAltNames}
Response	PEM-encoded certificate request
Errors	
Roles	Administrator

Request Properties:

- `type`: The string `csr`
- `keySize`: Key size, which must be either 2048 or 4096 bits. Optional property. Default value is 4096.


Note

This property is available as of version 2.15.0.

- `subjectDN`: The subject distinguished name (a string)
- `subjectAltNames`: Subject alternative names, as an array of objects with properties `type` and `value`. The value of property `type` is one of the following strings: IP, DNS, URI, or RFC822. The value of property `value` is the associated value (a string). This property is optional.

Example

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

Upload a Signed Certificate Chain as UI Certificate

Note

Available as of version 2.13.0.

URI	/api/v1/configuration/connector/ui/uiCertificate
Method	<i>PATCH</i>
Request	multipart/form-data with the following form parameter: signedCertificate
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- signedCertificate: The signed certificate and CA certificate chain (PEM-encoded)

Errors:

- INVALID_REQUEST (400): The certificate chain provided does not match the most recent certificate request, or it is not a certificate chain in the proper format (PEM-encoded).

Example

```
curl -i -k -u <user>:<password> -X PATCH -F signedCertificate=@<signedchain.pem>
https://<host>:<port>/api/v1/configuration/connector/ui/uiCertificate
```

Example

For test purposes, you can sign the certificate signing request with keytool.

```
keytool -genkeypair -keyalg RSA -keysize 1024 -alias mykey -dname "cn=very
trusted, c=test" -validity 365 -keystore ca.ks -keypass testit -storepass testit
keytool -gencert -rfc -infile csr.pem -outfile signedcsr.pem -alias mykey
-keystore ca.ks -keypass testit -storepass testit
keytool -exportcert -rfc -file ca.pem -alias mykey -keystore ca.ks -keypass
testit -storepass testit
cat signedcsr.pem ca.pem > signedchain.pem
```

Upload a PKCS#12 Certificate as UI Certificate

Note

Available as of version 2.13.0.

URI	/api/v1/configuration/connector/ui/ uiCertificat
Method	PUT
Request	multipart/form-data with the following form parameters: pkcs12 password keyPassword
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- `pkcs12`: Contents of PKCS#12 file
- `password`: Password for decrypting the PKCS#12 file
- `keyPassword`: Optional password for the private key

Errors:

- `INVALID_REQUEST (400)`: Contents of PKCS#12 or password are invalid

Note

keyPassword is optional. If missing, password is used to decrypt the pkcs#12 file and the private key.

Example

```
curl -i -k -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/ui/uiCertificate -X PUT -F 'password=<p12Password>' -F pkcs12=@<p12file>
```

Example

For test purposes, you can create an own self-signed pkcs#12 certificate with keytool.

```
keytool -genkeypair -alias key -keyalg RSA -keysize 2048 -validity 365 -keypass test20 -keystore test.p12 -storepass test20 -storetype PKCS12 -dname 'CN=test'
```

Get List of Available Cipher Suites for UI

Note

Available as of version 2.15.0.

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

URI	/api/v1/configuration/availableCipherSuites
Method	GET
Request	
Response	[name, ...]
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response Properties:

List of:

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

📘 Note

name is case-sensitive.

Example

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

Get List of Enabled Cipher Suites for UI

📘 Note

Available as of version 2.15.0.

Returns a list of enabled cipher suites.

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

Response Properties:

List of:

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

Note

name is case-sensitive.

Example

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

Set List of Enabled Cipher Suites for UI

Note

Available as of version 2.15.0.

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

Request Properties:

List of:

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

Note

name is case-sensitive.

Example

```
curl -i -k -H "Content-Type: application/json" -u <user>:<password> --  
data "[ 'TLS_AES_128_GCM_SHA256', 'TLS_AES_256_GCM_SHA384' ]" -X POST https://  
<host>:<port>/api/v1/configuration/connector/ui/cipherSuites
```

Enable default Cipher Suites for UI

📘 Note

Available as of version 2.15.0.

Revert changes and enable the default list of cipher suites.

URI	/api/v1/configuration/connector/ui/ cipherSuites
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Response Properties:

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

📘 Note

`name` is case-sensitive.

Example

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

1.5.2.5.5 Certificate Management for Backend Communication

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

Note

The APIs below are available as of Cloud Connector version 2.13.

There are two similar sets of APIs for system certificate and CA certificate for principal propagation.

Note

Some of the APIs list a parameter `subjectAltNames` (*subject alternative names* or SAN) for the request or response object. This parameter is an array of objects with the following properties:

- `type`: one of the strings *DNS*, *URI*, *IP*, or *RFC822*.
- `value`: the value associated with the chosen type.

- [CA Certificate for Principal Propagation: APIs \[page 396\]](#)
- [System Certificate: APIs \[page 402\]](#)
- [Truststore CA Certificates](#)

1.5.2.5.5.1 CA Certificate for Principal Propagation: APIs

Manage a CA certificate for principal propagation via API.

Note

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

Get Description for a CA Certificate for Principal Propagation

URI	<code>/api/v1/configuration/connector/ onPremise/ppCaCertificate</code>
Method	<code>GET</code>
Header	<code>Accept: application/json</code>

Response	<code>{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}</code>
Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response Properties:

- `subjectDN`: the subject distinguished name (a string)
- `issuer`: the issuer (a string)
- `notBeforeTimeStamp`: timestamp of the beginning of the validity period (a UTC long number)
- `notAfterTimeStamp`: timestamp of the end of the validity period (a UTC long number)
- `subjectAltNames`: subject alternative names (see [Certificate Management for Backend Communication \[page 396\]](#) for details).

Errors:

- NOT_FOUND (404): there is no CA certificate for principal propagation.

Note

`subjectAltNames` is not present if there are no subject alternative names.

Example

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

Get Binary Content of a CA Certificate for Principal Propagation

URI	<code>/api/v1/configuration/connector/onPremise/ppCaCertificate</code>
Method	<code>GET</code>
Header	<code>Accept: application/pkix-cert</code>
Response	Binary data of the certificate.

Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response:

- **Success:** the binary data of the certificate; you can verify the downloaded certificate by storing it in file `ppca.crt`, for instance, and then running

```
keytool -printcert -file ppca.crt
```

- **Failure:** an error in the usual JSON format; the content type of the response is *application/json* in this case.

Errors:

- **NOT_FOUND (404):** there is no CA certificate for principal propagation.

Example

```
curl -k -H "Accept: application/pkix-cert" -u <user>:<password> -X GET
--output sys.crt https://<host>:<port>/api/v1/configuration/connector/onPremise/
ppCaCertificate
```

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

URI	/api/v1/configuration/connector/ onPremise/ppCaCertificate
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.

Note

This property is available as of version 2.15.0.

- `subjectDN`: the subject distinguished name (a string)
- `subjectAltNames`: subject alternative names (see [Certificate Management for Backend Communication \[page 396\]](#) for details). This property is optional.

The certificate created this way has a validity of 1 year.

Example

```
curl -i -k -H "Accept: application/json" -H "Content-Type: application/json" -u <user>:<password> -X POST --data '{"type":"selfsigned", "subjectDN":"CN=me"}' https://<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate
```

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

URI	/api/v1/configuration/connector/onPremise/ppCaCertificate
Method	POST
Request	<pre>{type, keySize, subjectDN, subjectAltNames}</pre>
Response	PEM-encoded certificate request
Errors	
Roles	Administrator

Request Properties:

- `type`: the string *selfsigned*
- `keySize`: Key size, which must be either 2048 or 4096 bits. Optional property. Default value is 4096.

Note

This property is available as of version 2.15.0.

- `subjectDN`: the subject distinguished name (a string)

- `subjectAltNames`: subject alternative names (see [Certificate Management for Backend Communication \[page 396\]](#) for details). This property is optional.

Example

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

Upload a Signed Certificate Chain as CA Certificate for Principal Propagation (Master Only)

URI	/api/v1/configuration/connector/ onPremise/ppCaCertificate
Method	<i>PATCH</i>
Request	multipart/form-data with the following parameter: signedCertificate.
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `signedCertificate`: the signed certificate and CA certificate chain (PEM-encoded)

Errors:

- `INVALID_REQUEST` (400): the certificate chain provided does not match the most recent certificate request, or it is not a certificate chain in the correct format (PEM-encoded).

Example

```
curl -i -k -u <user>:<password> -X PATCH -F signedCertificate=@<signedchain.pem>
https://<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate
```

Example: Sign The Certificate Signing Request

```
keytool -genkeypair -keyalg RSA -keysize 1024 -alias mykey -dname "cn=very  
trusted, c=test" -validity 365 -keystore ca.ks -keypass testit -storepass testit  
keytool -gencert -rfc -infile csr.pem -outfile signedcsr.pem -alias mykey  
-keystore ca.ks -keypass testit -storepass testit  
keytool -exportcert -rfc -file ca.pem -alias mykey -keystore ca.ks -keypass  
testit -storepass testit  
cat signedcsr.pem ca.pem > signedchain.pem
```

Upload a PKCS#12 Certificate as CA Certificate for Principal Propagation (Master Only)

URI	/api/v1/configuration/connector/ onPremise/ppCaCertificate
Method	PUT
Request	Multipart/form-data with the following form parameters: pkcs12, password, keyPassword
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- `pkcs12`: contents of PKCS#12 file
- `password`: password for decrypting PKCS#12 file
- `keyPassword`: optional password for the private key

Errors:

- `INVALID_REQUEST (400)`: contents of PKCS#12 or password are invalid

Note

`keyPassword` is optional. If it is missing, `password` is used to decrypt the `pkcs#12` file and the private key.

Example

```
curl -i -k -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate -X PUT -F 'password=<p2lPassword>' -F pkcs12=@<p12file>
```

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

```
keytool -genkeypair -alias key -keyalg RSA -keysize 2048 -validity 365 -keypass test20 -keystore test.p12 -storepass test20 -storetype PKCS12 -dname 'CN=test'
```

Delete a CA Certificate for Principal Propagation (Master Only)

URI	/api/v1/configuration/connector/onPremise/ppCaCertificate
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator

Errors:

- NOT_FOUND (404): there is no CA certificate for principal propagation.

Example

```
curl -k -H "Accept: application/json" -u <user>:<password> --request DELETE https://localhost:8443/api/v1/configuration/connector/onPremise/ppCaCertificate
```

1.5.2.5.5.2 System Certificate: APIs

Manage a system certificate via API.

Note

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

Get Description for a System Certificate

URI	/api/v1/configuration/connector/ onPremise/systemCertificate
Method	GET
Header	Accept: application/json
Response	<pre>{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}</pre>
Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response Properties:

- `subjectDN`: the subject distinguished name (a string)
- `issuer`: the issuer (a string)
- `notBeforeTimeStamp`: timestamp of the beginning of the validity period (a UTC long number)
- `notAfterTimeStamp`: timestamp of the end of the validity period (a UTC long number)
- `subjectAltNames`: subject alternative names.

Errors:

- `NOT_FOUND` (404): there is no system certificate.

Note

`subjectAltNames` is not present if there are no subject alternative names.

Example

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

Get Binary Content of a System Certificate

URI	/api/v1/configuration/connector/ onPremise/systemCertificate
Method	GET
Header	Accept: application/pkix-cert
Response	Binary data of the certificate.
Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response:

- **Success:** the binary data of the certificate; you can verify the downloaded certificate by storing it in file `sys.crt`, for instance, and then running

```
keytool -printcert -file sys.crt
```

- **Failure:** an error in the usual JSON format; the content type of the response is *application/json* in this case.

Errors:

- **NOT_FOUND (404):** there is no system certificate.

Example

```
curl -i -k -H "Accept: application/pkix-cert" -u <user>:<password> -X GET  
--output sys.crt https://<host>:<port>/api/v1/configuration/connector/  
onPremise/systemCertificate
```

Create a Self-Signed System Certificate (Master Only)

URI	/api/v1/configuration/connector/ onPremise/systemCertificate
Method	POST
Header	CONTENT_TYPE: application/json

Request	<code>{type, keySize, subjectDN}</code>
Response	201 on success
Errors	
Roles	Administrator

Request Properties:

- `type`: the string *selfsigned*
- `keySize`: Key size, which must be either 2048 or 4096 bits. Optional property. Default value is 4096.

Note

This property is available as of version 2.15.0.

- `subjectDN`: the subject distinguished name (a string)
- `subjectAltNames`: subject alternative names. This property is optional. As of version 2.16.1, the value is ignored. SAN on system certificate had never any effect.

The certificate created this way has a validity of 1 year.

Example

```
curl -i -k -H "Accept: application/json" -H "Content-Type: application/json" -u <user>:<password> -X POST --data '{"type":"selfsigned", "subjectDN":"CN=me"}' https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate
```

Create a Certificate Signing Request for a System Certificate (Master Only)

URI	<code>/api/v1/configuration/connector/onPremise/systemCertificate</code>
Method	<i>POST</i>
Header	CONTENT_TYPE: application/json
Request	<code>{type, keySize, subjectDN}</code>
Response	PEM-encoded certificate request

Errors

Roles

Administrator

Request Properties:

- `type`: the string `csr`
- `keySize`: Key size, which must be either 2048 or 4096 bits. Optional property. Default value is 4096.

Note

This property is available as of version 2.15.0.

- `subjectDN`: the subject distinguished name (a string)
- `subjectAltNames`: subject alternative names. This property is optional. As of version 2.16.1, the value is ignored. SAN on system certificate had never any effect.

Example

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

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

URI	/api/v1/configuration/connector/onPremise/systemCertificate
Method	PATCH
Request	multipart/form-data with the following parameter: <code>signedCertificate</code> .
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `signedCertificate`: the signed certificate and CA certificate chain (PEM-encoded)

Errors:

- **INVALID_REQUEST (400):** the certificate chain provided does not match the most recent certificate request, or it is not a certificate chain in the correct format (PEM-encoded).

Example

```
curl -i -k -u <user>:<password> -X PATCH -F signedCertificate=@<signedchain.pem> https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate
```

For test purposes, you can sign the certificate signing request with keytool:

```
keytool -genkeypair -keyalg RSA -keysize 1024 -alias mykey -dname "cn=very trusted, c=test" -validity 365 -keystore ca.ks -keypass testit -storepass testit
keytool -gencert -rfc -infile csr.pem -outfile signedcsr.pem -alias mykey -keystore ca.ks -keypass testit -storepass testit
keytool -exportcert -rfc -file ca.pem -alias mykey -keystore ca.ks -keypass testit -storepass testit
cat signedcsr.pem ca.pem > signedchain.pem
```

Upload a PKCS#12 Certificate as System Certificate (Master Only)

URI	/api/v1/configuration/connector/onPremise/systemCertificate
Method	<i>PUT</i>
Request	Multipart/form-data with the following form parameters: <code>pkcs12</code> , <code>password</code> , <code>keyPassword</code>
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- `pkcs12`: contents of PKCS#12 file
- `password`: password for decrypting PKCS#12 file
- `keyPassword`: optional password for the private key

Errors:

- `INVALID_REQUEST (400)`: contents of PKCS#12 or password are invalid

Note

`keyPassword` is optional. If it is missing, `password` is used to decrypt the pkcs#12 file and the private key.

Example

```
curl -i -k -u <user>:<password> https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate -X PUT -F password=<p21Password> -F pkcs12=@<p12file>
```

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

```
keytool -genkeypair -alias key -keyalg RSA -keysize 2048 -validity 365 -keypass test20 -keystore test.p12 -storepass test20 -storetype PKCS12 -dname 'CN=test'
```

Delete a System Certificate (Master Only)

URI	/api/v1/configuration/connector/onPremise/systemCertificate
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator

Errors:

- `NOT_FOUND (404)`: there is no system certificate.

Example

```
curl -k -H "Accept: application/json" -u <user>:<password> --request DELETE https://localhost:8443/api/v1/configuration/connector/onPremise/systemCertificate
```

1.5.2.5.6 Solution Management Configuration

Manage the Cloud Connector's solution management configuration via API.

Get Solution Management Configuration

URI	/api/v1/configuration/connector/ solutionManagement
Method	GET
Request	
Response	{hostAgentPath, isEnabled, dsrEnabled}
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `isEnabled`: flag indicating if reporting to solution management is active.
- `hostAgentPath`: path for host agent executable.
- `dsrEnabled`: indicating if DSR reporting is active.

Example

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

Set Solution Management Configuration and Turn On Reporting

This API turns on the integration with the Solution Manager. The prerequisite is an available Host Agent. You can specify a path to the Host Agent executable, if you don't use the default path.

URI	/api/v1/configuration/connector/ solutionManagement
Method	POST
Request	{hostAgentPath, dsrEnabled}
Response	
Errors	
Roles	Administrator, Support

Response Properties:

- hostAgentPath: path for host agent executable (string, optional).
- dsrEnabled: flag indicating if DSR reporting is active (boolean, optional)-

Example

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

or, if configuration has to be changed:

```
curl -ik -u <user>:<password> https://<host>:<port>/api/v1/configuration/  
connector/solutionManagement -X POST -d '{"hostAgentPath":"new/path/to/  
hostAgent\"}" -H "Content-Type:application/json"
```

Turn Off Solution Management Reporting

URI	/api/v1/configuration/connector/ solutionManagement
-----	--

Method	<i>DELETE</i>
Request	
Response	
Errors	
Roles	Administrator, Support

Example

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

Download Current LMDB XML Report

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

Note

Available as of Cloud Connector version 2.12.0.

URI	/api/v1/configuration/connector/solutionManagement/registrationFile
Method	<i>GET</i>
Request	
Response	
Errors	
Roles	Administrator, Support

1.5.2.5.7 Backup

Manage the Cloud Connector's configuration backup via API.

Create Backup Configuration

URI	/api/v1/configuration/backup
Method	POST
Request	{password}
Response	ZIP archive (content type application/zip)
Errors	
Roles	Administrator

Request Properties:

- `password`: the password used to encrypt sensitive data.

Note

Only sensitive data in the backup are encrypted with an arbitrary password of your choice. The password is required for the restore operation. The returned ZIP archive itself is not password-protected.

Sample Code

```
curl -k -u Administrator:<password> https://<host>:<port>/api/v1/
configuration/backup -X POST -H 'Content-Type: application/json' -d
"{\"password\": \"<password>\"}" -o <backupfile>.zip
```

Restore Backup Configuration

Caution

A successful request triggers a restart of the Cloud Connector.

URI	/api/v1/configuration/backup
Method	PUT
Request	multipart/form-data with the following form parameters: backup, password.
Response	204 on success
Errors	INVALID_REQUEST
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.

📌 Note

Since this API uses a multipart request, it requires a multipart request header.

🔗 Sample Code

```
curl -k -u Administrator:<password> https://<host>:<port>/api/v1/
configuration/backup -X PUT -F 'password=<password>' -F
backup=@<backupfile>.zip
```

1.5.2.5.8 Subaccount

Manage the Cloud Connector's subaccount settings via API.

Operations

Subaccount

[Get Subaccounts \[page 414\]](#)

[Create Subaccount \(Master Only\) \[page 415\]](#)

[Delete Subaccount \(Master Only\) \[page 416\]](#)

[Edit Subaccount \(Master Only\) \[page 416\]](#)

[Connect/Disconnect Subaccount \(Master Only\) \[page 417\]](#)

[Refresh Subaccount Certificate \(Master Only\) \[page 418\]](#)

[Get Subaccount Configuration \[page 419\]](#)

Recovery Subaccount

⚠ Caution

This feature is deprecated.

Due to the discontinuation of the *Enhanced Disaster Recovery Service*, the related functionality in the Cloud Connector has been dropped as of version 2.16.

For more information, see [What's New for SAP Business Technology Platform](#).

[Create Recovery Subaccount \(Master Only\) \[page 420\]](#)

[Delete Recovery Subaccount \(Master Only\) \[page 420\]](#)

[Refresh Certificate of Recovery Subaccount \(Master Only\) \[page 420\]](#)

[Activate/Deactivate Recovery Subaccount \(Master Only\) \[page 420\]](#)

[Takeover \(Master Only\) \[page 421\]](#)

Trust

[Synchronize Trust List \(Master Only\) \[page 421\]](#)

[Get IDP Trust List \[page 422\]](#)

[Get Application Trust List \[page 422\]](#)

[Enable Or Disable Trust \(Master Only\) \[page 423\]](#)

[Get IDP Trust Properties \[page 424\]](#)

[Get Application Trust Properties \[page 424\]](#)

Get Subaccounts

URI	<code>/api/v1/configuration/subaccounts</code>
-----	--

Method	<code>GET</code>
--------	------------------

Request	
---------	--

Response	<pre>[{regionHost, subaccount, locationID}]</pre>
----------	---

Errors	
--------	--

Response:

An array of objects with the following properties:

- `regionHost`: region hosts (a string).
- `subaccount`: subaccount name (a string).
- `locationID`: location identifier for the Cloud Connector instance (a string); this property is not available if the default location ID is in use.

Back to [Operations \[page 413\]](#)

Create Subaccount (Master Only)

Creates and connects a subaccount.

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

Back to [Operations \[page 413\]](#)

Delete Subaccount (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount></code>
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 \[page 413\]](#)

Edit Subaccount (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>
Method	PUT
Request	{locationID, displayName, description}
Response	{regionHost, subaccount, locationID, displayName, description, tunnel}
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.

Errors:

- **NOT_FOUND (404):** subaccount does not exist (in the specified region).

Back to [Operations \[page 413\]](#)

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 \[page 413\]](#)

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}
Errors	
Roles	Administrator, Subaccount Administrator

Request Properties:

- `user`: user for the specified region host and subaccount.
- `password`: password for the (cloud) user.

Response Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).

- `locationID`: location identifier for the Cloud Connector instance (a string); this property is not available if the default location ID is in use.
- `displayName`: display name of the subaccount (a string); this property is not available if there is no specified display name.
- `description`: subaccount description (a string); this property is not available if there is no description.
- `tunnel`: object outlining the current state of the tunnel.

Back to [Operations \[page 413\]](#)

Get Subaccount Configuration

URI	<code>/api/v1/configuration/subaccounts/<regionHost>/<subaccount></code>
Method	<i>GET</i>
Request	
Response	<pre>{regionHost, subaccount, locationID, displayName, description, tunnel:{state, connections, applicationConnections: [], serviceChannels:[]}}</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).
- `locationID`: location ID (a string); this property is not available if the default location ID is in use.
- `displayName`: display name of the subaccount (a string); this property is not available if there is no specified display name.
- `description`: description (a string); this property is not available if there is no description.
- `tunnel`: array of connection tunnels used by the subaccount.
 - `state`: *Connected*, *ConnectFailure*, or *Disconnected*
 - `connectedSinceTimeStamp`: connection start time as UTC timestamp
 - `connections`: number of subaccount connections
 - `applicationConnections`: array of connections to application instances
 - `serviceChannels`: type and state of the service channels used (types: HANA database, Virtual Machine or RFC)

- `subaccountCertificate`: information on the subaccount certificate such as validity period, issuer and subject DN

Back to [Operations \[page 413\]](#)

Create Recovery Subaccount (Master Only)

⚠ Caution

Disaster recovery discontinued as of version 2.16. A `POST` request to `/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery` will trigger a 410 response.

Back to [Operations \[page 413\]](#)

Delete Recovery Subaccount (Master Only)

⚠ Caution

Disaster recovery discontinued as of version 2.16. A `DELETE` request to `/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery` will trigger a 410 response.

Back to [Operations \[page 413\]](#)

Refresh Certificate of Recovery Subaccount (Master Only)

⚠ Caution

Disaster recovery discontinued as of version 2.16. A `POST` request to `/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/validity` will trigger a 410 response.

Back to [Operations \[page 413\]](#)

Activate/Deactivate Recovery Subaccount (Master Only)

⚠ Caution

Disaster recovery discontinued as of version 2.16. A PUT request to `/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/state` will trigger a 410 response.

Back to [Operations \[page 413\]](#)

Takeover (Master Only)

⚠ Caution

Disaster recovery discontinued as of version 2.16. A POST request to `/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/takeover` will trigger a 410 response.

Back to [Operations \[page 413\]](#)

Synchronize Trust List (Master Only)

📌 Note

Available as of version 2.14.0.

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

Back to [Operations \[page 413\]](#)

Get IDP Trust List

Note

Available as of version 2.14.0.

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/trust/idps</code>
Method	<code>GET</code>
Request	
Response	<code>[{id, name, description, certificate, enabled}]</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each of which represents a trusted IDP through the following properties:

- `id`: (unique) ID of the trusted IDP (a number).
- `name`: name of the trusted IDP (a string).
- `description`: description of the trusted IDP (a string)
- `certificate`: object with the following properties: `issuer` (a string), `subjectDN` (a string), `notBeforeTimeStamp` (a UTC long number), and `notAfterTimeStamp` (a UTC long number).
- `enabled`: flag that indicates whether the IDP is enabled or disabled (that is, whether the IDP is trusted or not).

Back to [Operations \[page 413\]](#)

Get Application Trust List

Note

Available as of version 2.14.0.

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/trust/apps
Method	GET
Request	
Response	<code>[{id, name, applicationType, enabled}]</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each of which represents a trusted application through the following properties:

- `id`: (unique) ID of the trusted application (a number).
- `name`: name of the trusted application (a string).
- `applicationType`: type of the application (a string, for example 'JAVA' or 'HANA').
- `enabled`: flag that indicates whether the application is enabled or disabled (that is, whether the application is trusted or not).

Back to [Operations \[page 413\]](#)

Enable Or Disable Trust (Master Only)

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

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 \[page 413\]](#)

Get IDP Trust Properties

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/trust/idps/ <id></code>
Method	<code>GET</code>
Request	
Response	<code>{id, name, description, certificate, enabled}</code>
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Response Properties:

- `id`: (unique) ID of the trusted IDP (a number).
- `name`: name of the trusted IDP (a string).
- `description`: description of the trusted IDP (a string).
- `certificate`: object with the following properties: `issuer` (a string), `subjectDN` (a string), `notBeforeTimeStamp` (a UTC long number), and `notAfterTimeStamp` (a UTC long number).
- `enabled`: flag that indicates whether the IDP is enabled or disabled (that is, whether the IDP is trusted or not).

Errors:

- NOT_FOUND (404): There is no trusted IDP with the given `<id>`

Back to [Operations \[page 413\]](#)

Get Application Trust Properties

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

Request	
Response	<code>{id, name, applicationType, enabled}</code>
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Response Properties:

- `id`: (unique) ID of the trusted application (a number).
- `name`: name of the trusted application (a string).
- `applicationType`: type of the application (a string, for example 'JAVA' or 'HANA').
- `enabled`: flag that indicates whether the application is enabled or disabled (that is, whether the application is trusted or not).

Errors:

- NOT_FOUND (404): There is no trusted application with the given <id>.

Back to [Operations \[page 413\]](#)

1.5.2.5.9 System Mappings

Manage the Cloud Connector's system mappings via API.

Get All System Mappings

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMappings</code>
Method	<code>GET</code>
Request	
Response	<code>[{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader, description, ...}]</code>
Errors	

Response:

An array of objects with the following properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)
- `localHost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses (a string)
- `backendType`: Type of the backend (a string)
- `authenticationMode`: Authentication mode used on the backend side (a string).
- `hostInHeader`: Policy for setting the host in the response header. Available for HTTP(S) protocols only.
- `totalResourcesCount`: the total number of resources
- `enabledResourcesCount`: the number of enabled resources
- `description`: Description for the system mapping (a string).
- `sncPartnerName`: SNC name of an ABAP Server, required for RFCS communication only.
- `sapRouter`: SAP router route, required only if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedUsers`: Array of {client, user}, describing users that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

Get System Mapping

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/ <virtualHost>:<virtualPort></code>
Method	<code>GET</code>
Request	
Response	<code>{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader, description, ...}</code>
Errors	

Response:

An array of objects with the following properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)
- `localhost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses (a string)
- `backendType`: Type of the backend (a string)
- `authenticationMode`: Authentication mode used on the backend side (a string).
- `hostInHeader`: Policy for setting the host in the response header (a string). Available for HTTP(S) protocols only.
- `totalResourcesCount`: the total number of resources
- `enabledResourcesCount`: the number of enabled resources
- `description`: Description for the system mapping (a string).
- `sncPartnerName`: SNC name of an ABAP Server, required for RFCS communication only.
- `sapRouter`: SAP router route, required only if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedUsers`: Array of {client, user}, describing users that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

Create System Mapping (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMappings
Method	POST
Request	<pre>{virtualHost, virtualPort, localhost, localhost, protocol, backendType, authenticationMode, hostInHeader, description, ...}</pre>
Response	201 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request Properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)
- `localhost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses, which must be one of the following strings: `HTTP`, `HTTPS`, `RFC`, `RFCS`, `LDAP`, `LDAPS`, `TCP`, `TCPs`.
- `backendType`: Type of the backend system. Valid values are `abapSys`, `netweaverCE`, `netweaverGW`, `applServerJava`, `BC`, `PI`, `hana`, `otherSAPsys`, `nonSAPsys`.
- `authenticationMode`: Authentication mode to be used on the backend side, which must be one of the following strings: `NONE`, `NONE_RESTRICTED`, `X509_GENERAL`, `X509_RESTRICTED`, `KERBEROS`.
- `hostInHeader`: Policy for setting the host in the response header. This property is applicable to HTTP(S) protocols only, and it is **optional**. If set, it must be one of the following strings: `internal`, `virtual`. The default is `virtual`. You may also use all capital letters, i.e. `INTERNAL` and `VIRTUAL`.
- `description`: Description for the system mapping (string, optional). The default is no description, i.e. the empty string.
- `sncPartnerName`: SNC name of an ABAP Server, required for RFCS communication only.
- `sapRouter`: SAP router route, required only if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedUsers`: Array of `{client, user}`, describing users that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

Note

The authentication modes `NONE_RESTRICTED` and `X509_RESTRICTED` prevent the Cloud Connector from sending the system certificate in any case, whereas `NONE` and `X509_GENERAL` will send the system certificate if the circumstances allow it.

Delete System Mapping (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMappings/ <virtualHost>:<virtualPort></code>
Method	<code>DELETE</code>
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	DELETE
--------	--------

Request	
---------	--

Response	204 on success
----------	----------------

Errors

Roles	Administrator, Subaccount Administrator
-------	---

Replace System Mapping (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort
-----	--

Method	PUT
--------	-----

Request	<pre>{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader, description, ...}</pre>
---------	--

Response	204 on success
----------	----------------

Errors

Roles	Administrator, Subaccount Administrator
-------	---

Request Properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)

- `localhost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses, which must be one of the following strings: `HTTP`, `HTTPS`, `RFC`, `RFCS`, `LDAP`, `LDAPS`, `TCP`, `TCPS`.
- `backendType`: Type of the backend system. Valid values are `abapSys`, `netweaverCE`, `netweaverGW`, `applServerJava`, `BC`, `PI`, `hana`, `otherSAPsys`, `nonSAPsys`.
- `authenticationMode`: Authentication mode to be used on the backend side, which must be one of the following strings: `NONE`, `NONE_RESTRICTED`, `X509_GENERAL`, `X509_RESTRICTED`, `KERBEROS`.
- `hostInHeader`: Policy for setting the host in the response header; this property is **optional**. If set, it must be one of the following strings: `internal`, `virtual`. The default is `virtual`. You may also use all capital letters, i.e. `INTERNAL` and `VIRTUAL`.
- `description`: Description for the system mapping (string, optional). The default is no description, i.e. the empty string.
- `sncPartnerName`: SNC name of an ABAP Server, only set for RFCS communication.
- `sapRouter`: SAP router route, only set if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedUsers`: Array of `{client, user}`, describing users, that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

📌 Note

The authentication modes `NONE_RESTRICTED` and `X509_RESTRICTED` prevent the Cloud Connector from sending the system certificate in any case, whereas `NONE` and `X509_GENERAL` will send the system certificate if the circumstances allow it.

1.5.2.5.10 System Mapping Resources

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

Operations

System Mapping Resources

System Mapping Resources	Get all System Mapping Resources [page 431]
	Get System Mapping Resource [page 432]
	Create System Mapping Resource [page 433]
	Edit System Mapping Resource [page 434]
	Delete System Mapping Resource [page 434]
	Delete all System Mapping Resources [page 435]
Allowed Clients	Get Allowed Clients for RFC System Mapping [page 435]
	Set Allowed Clients for RFC System Mapping [page 436]
	Add Allowed Clients for RFC System Mapping [page 437]
	Delete an Allowed Client for RFC System Mapping [page 438]
	Delete all Allowed Clients for RFC System Mapping [page 438]
Blocked Users	Get Blocked Users for RFC System Mapping [page 439]
	Set Blocked Users for RFC System Mapping [page 440]
	Add Blocked Users for RFC System Mapping [page 440]
	Remove one Blocked User for RFC System Mapping [page 441]
	Remove all Blocked Users for RFC System Mapping [page 442]

Get all System Mapping Resources

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort/ resources</code>
Method	<code>GET</code>
Request	
Response	<code>[{id, enabled, exactMatchOnly, websocketUpgradeAllowed, description}]</code>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each representing a resource through the following properties:

- **id**: The resource itself, which, depending on the owning system mapping, is either a URL path (or the leading section of it), or a RFC function name (prefix)
- **enabled**: Boolean flag indicating whether the resource is enabled.
- **exactMatchOnly**: Boolean flag determining whether access is granted only if the requested resource is an exact match.
- **websocketUpgradeAllowed**: Boolean flag indicating whether websocket upgrade is allowed; this property is of relevance only if the owning system mapping employs protocol HTTP or HTTPS.
- **description**: Description (a string); this property is not available unless explicitly set.

Back to [Top \[page 430\]](#)

Get System Mapping Resource

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort/ resources/<encodedResourceId></code>
Method	<i>GET</i>
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 \[page 430\]](#)

Create System Mapping Resource

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort/ resources</code>
Method	<code>POST</code>
Request	<code>{id, enabled, exactMatchOnly, websocketUpgradeAllowed, description}</code>
Response	201 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request Properties:

- `id`: The resource itself, which, depending on the owning system mapping, is either a URL path (or the leading section of it), or a RFC function name (prefix).
- `enabled`: Boolean flag indicating whether the resource is enabled (**optional**). The default value is `false`.
- `exactMatchOnly`: Boolean flag determining whether access is granted only if the requested resource is an exact match (**optional**). The default value is `false`.
- `websocketUpgradeAllowed`: Boolean flag indicating whether websocket upgrade is allowed (**optional**). The default value is `false`. This property is recognized only if the owning system mapping employs protocol HTTP or HTTPS.
- `description`: Description (a string, **optional**)

→ Tip

Encoded Resource ID

URI paths may contain the resource ID in order to identify the resource to be edited or deleted. A resource ID, however, may contain characters such as the forward slash that collide with the path separator of the URI and hence require an escape mechanism. We adopted the following simple escape or encoding method for a resource ID:

1. Replace all occurrences of character '+' with '+2B'.
2. Replace all occurrences of character '-' with '+2D'.
3. Replace all occurrences of character '/' with '-'

Back to [Top \[page 430\]](#)

Replace System Mapping Resource

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort/ resources/<encodedResourceId></code>
Method	<i>PUT</i>
Request	<pre>{enabled, exactMatchOnly, websocketUpgradeAllowed, description}</pre>
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request Properties:

- `enabled`: Boolean flag indicating whether the resource is enabled.
- `exactMatchOnly`: Boolean flag determining whether access is granted only if the requested resource is an exact match.
- `websocketUpgradeAllowed`: Boolean flag indicating whether websocket upgrade is allowed; this property is of relevance only if the owning system mapping employs protocol HTTP or HTTPS.
- `description`: Description (a string); this property is not available unless explicitly set.

Back to [Top \[page 430\]](#)

Delete System Mapping Resource

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ systemMappings/virtualHost:virtualPort/ resources/<encodedResourceId></code>
Method	<i>DELETE</i>
Request	
Response	204 on success

Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Errors:

NOT_FOUND (404): Resource was not found

Back to [Top \[page 430\]](#)

Delete all System Mapping Resources

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

Back to [Top \[page 430\]](#)

Get Allowed Clients for RFC System Mapping

Note

Available as of version 2.1.4.

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

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

Response Properties:

- Array of allowed clients:
 - `client`: ABAP client

Note

An empty list means that every client is allowed.

Back to [Top \[page 430\]](#)

Set Allowed Clients for RFC System Mapping

Note

Available as of version 2.1.4.

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

Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Request Properties:

- Array of allowed clients:
 - `client`: ABAP client

The existing list of available clients will be cleaned and populated by the provided in request.

Back to [Top \[page 430\]](#)

Add Allowed Clients for RFC System Mapping

Note

Available as of version 2.1.4.

Adds more ABAP clients to the allowed list for the system mapping definition.

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

Request Properties:

- Array of allowed clients:
 - `client`: ABAP client

Clients provided in this request will be added to the existing list. Clients that are already listed will be ignored.

Back to [Top \[page 430\]](#)

Delete an Allowed Client for RFC System Mapping

Note

Available as of version 2.1.4.

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

Back to [Top \[page 430\]](#)

Delete all Allowed Clients for RFC System Mapping

Note

Available as of version 2.1.4.

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

Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

📘 Note

An empty list means that every client is allowed.

Back to [Top \[page 430\]](#)

Get Blocked Users for RFC System Mapping

📘 Note

Available as of version 2.1.4.

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

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

Response Properties:

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

Back to [Top \[page 430\]](#)

Set Blocked Users for RFC System Mapping

Note

Available as of version 2.1.4.

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

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

Request Properties:

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

Back to [Top \[page 430\]](#)

Add Blocked Users for RFC System Mapping

Note

Available as of version 2.1.4.

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

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

Request Properties:

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

Back to [Top \[page 430\]](#)

Remove one Blocked User for RFC System Mapping

Note


Available as of version 2.1.4.

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

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

Back to [Top \[page 430\]](#)

Remove all Blocked Users for RFC System Mapping

 **Note**
Available as of version 2.1.4.

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

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

Back to [Top \[page 430\]](#)

1.5.2.5.11 Domain Mappings

Manage the Cloud Connector's configuration for domain mappings via API.

Get Domain Mappings

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/domainMappings
Method	GET
Request	
Response	[{virtualDomain, internalDomain}]
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each representing a domain mapping through the following properties:

- virtualDomain: Domain used on the cloud side
- internalDomain: Domain used on the on-premise side

Create Domain Mapping (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/domainMappings
Method	POST
Request	{virtualDomain, internalDomain}
Response	201
Errors	
Roles	Administrator, Subaccount Administrator

Replace Domain Mapping (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ domainMappings/<internalDomain>
-----	---

Method	<i>PUT</i>
Request	<code>{virtualDomain, internalDomain}</code>
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Request:

- `virtualDomain`: New virtual domain
- `internalDomain`: New internal domain

Errors:

- NOT_FOUND (404): Domain mapping does not exist.

Note

The internal domain in the URI path (i.e., `<internalDomain>`) is the current internal domain of the domain mapping that is to be edited. It may differ from the new internal domain set in the request.

Delete Domain Mapping (Master Only)

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

Errors:

- NOT_FOUND (404): Domain mapping does not exist.

Delete All Domain Mappings (Master Only)

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

1.5.2.5.12 Subaccount Service Channels

Manage Cloud Connector service channels via API.

Service Channel for HANA Database	Get all HANA Service Channels [page 446]
	Get Available HANA Channels [page 447]
	Get HANA Service Channel [page 448]
	Create HANA Service Channel (Master Only) [page 449]
	Replace HANA Service Channel (Master Only) [page 449]
Service Channel for Virtual Machine	Get all Service Channels for Virtual Machines [page 450]
	Get Available Channels for Virtual Machines [page 451]
	Get Service Channel for a Virtual Machine [page 451]
	Create a Service Channel for a Virtual Machine (Master Only) [page 452]
	Replace a Service Channel for a Virtual Machine (Master Only) [page 453]
Service Channel for ABAP Cloud	Get all ABAP Cloud Service Channels [page 454]
	Get ABAP Cloud Service Channel [page 455]
	Create ABAP Cloud Service Channel (Master Only) [page 455]
	Replace ABAP Cloud Service Channel (Master Only) [page 456]

Service Channel for Kubernetes

Note

Available as of version 2.15.0.

[Get All Kubernetes Cluster Service Channels \[page 457\]](#)

[Get Kubernetes Service Channel \[page 458\]](#)

[Create Kubernetes Service Channel \(Master Only\) \[page 459\]](#)

[Replace Kubernetes Service Channel \(Master Only\) \[page 459\]](#)

General

[Enable/Disable Service Channel \(Master Only\) \[page 460\]](#)

[Delete Service Channel \(Master Only\) \[page 461\]](#)

[Delete all Service Channels \(Master Only\) \[page 461\]](#)

Deprecated

[Get all Service Channels \[page 462\]](#)

[Get Service Channel \[page 463\]](#)

[Create Service Channel \[page 464\]](#)

[Replace Service Channel \[page 464\]](#)

Get all HANA Service Channels

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/HANA</code>
Method	<code>GET</code>
Request	
Response	<pre>[{id, hanaInstanceName, instanceNumber, type, port, enabled, connections, state}]</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each of which represents a HANA service channel through the following properties:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1). This identifier is unique across all types of service channels.
- `hanaInstanceName`: name of the HANA instance (a string).
- `instanceNumber`: instance number.
- `type`: string 'HANA'.

- **port**: port of the HANA service channel (a number).
- **enabled**: boolean flag indicating whether the channel is enabled and therefore should be open.
- **connections**: maximal number of open connections.
- **state**: current connection state; this property is only available if the channel is enabled (as per property **enabled**). The value of this property is an object with the properties:
 - **connected** (a boolean flag indicating whether the channel is connected),
 - **openedConnections** (the number of open, possibly idle connections), and
 - **connectedSinceTimeStamp** (the time stamp, a UTC long number, for the first time the channel was opened/connected).

Back to [Top \[page 445\]](#)

Get Available HANA Channels

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

Back to [Top \[page 445\]](#)

Get HANA Service Channel

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/HANA/ <id></code>
Method	<i>GET</i>
Request	
Response	<pre>{id, hanaInstanceName, instanceNumber, type, port, enabled, connections, state}</pre>
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 \[page 445\]](#)

Create HANA Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/HANA
Method	POST
Request	<pre>{hanaInstanceName, instanceNumber, connections}</pre>
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator, Subaccount Administrator

Request:

- `hanaInstanceName`: name of the HANA instance (a string).
- `instanceNumber`: instance number.
- `connections`: maximal number of open connections.

Errors:

- INVALID_REQUEST (400): invalid or out of range values.

Back to [Top \[page 445\]](#)

Replace HANA Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/HANA/ <id>
Method	PUT
Request	<pre>{hanaInstanceName, instanceNumber, connections}</pre>
Response	204 on success
Errors	INVALID_REQUEST, NOT_FOUND
Roles	Administrator, Subaccount Administrator

Request:

- `hanaInstanceName`: name of the HANA instance (a string).
- `instanceNumber`: instance number.
- `connections`: maximal number of open connections.

Errors:

- `INVALID_REQUEST` (400): invalid or out of range values.
- `NOT_FOUND` (404): the HANA service channel with the given ID (that is, `<id>`) or the specified subaccount does not exist.

Back to [Top \[page 445\]](#)

Get all Service Channels for Virtual Machines

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ VirtualMachine</code>
Method	<code>GET</code>
Request	
Response	<pre>[{id, name, endpointId, type, port, enabled, connections, state}]</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1). This identifier is unique across all types of service channels.
- `name`: name of the virtual machine on the cloud platform (a string).
- `endpointId`: unique ID (GUID) of the virtual machine (a string).
- `type`: string 'VirtualMachine'.
- `port`: port of the service channel for the virtual machine (a number).
- `enabled`: boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.
- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties:
 - `connected` (a boolean flag indicating whether the channel is connected),
 - `openedConnections` (the number of open, possibly idle connections), and

- `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).

Back to [Top \[page 445\]](#)

Get Available Channels for Virtual Machines

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/ availableChannels/VirtualMachine</code>
Method	<i>GET</i>
Request	
Response	<code>[{endpointId, name}]</code>
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 \[page 445\]](#)

Get Service Channel for a Virtual Machine

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ VirtualMachine/<id></code>
Method	<i>GET</i>
Request	

Response	<code>{id, name, endpointId, type, port, enabled, connections, state}</code>
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- **id**: unique identifier for the service channel (a positive integer number, starting with 1). This identifier is unique across all types of service channels.
- **name**: name of the virtual machine on the cloud platform (a string).
- **endpointId**: unique ID (GUID) of the virtual machine (a string).
- **type**: string 'VirtualMachine'.
- **port**: port of the service channel for the virtual machine (a number).
- **enabled**: boolean flag indicating whether the channel is enabled and therefore should be open.
- **connections**: maximal number of open connections.
- **state**: current connection state; this property is only available if the channel is enabled (as per property **enabled**). The value of this property is an object with the properties:
 - **connected** (a boolean flag indicating whether the channel is connected),
 - **openedConnections** (the number of open, possibly idle connections), and
 - **connectedSinceTimeStamp** (the time stamp, a UTC long number, for the first time the channel was opened/connected).

Errors:

- NOT_FOUND (404): the service channel for a virtual machine with the given ID (that is, <id>) or the specified subaccount does not exist.

Back to [Top \[page 445\]](#)

Create a Service Channel for a Virtual Machine (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ VirtualMachine</code>
Method	<i>POST</i>
Request	<code>{name, endpointId, port, connections}</code>
Response	201 on success

Errors	INVALID_REQUEST
Roles	Administrator, Subaccount Administrator

Request:

- `name`: name of the virtual machine on the cloud platform (a string).
- `endpointId`: unique ID (GUID) of the virtual machine (a string).
- `port`: port of the service channel for the virtual machine (a number).
- `connections`: maximal number of open connections.

Errors:

- INVALID_REQUEST (400): invalid or out of range values.

Back to [Top \[page 445\]](#)

Replace Service Channel for a Virtual Machine (Master Only)

URI	<code>/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/VirtualMachine/<id></code>
Method	<i>PUT</i>
Request	<code>{name, endpointId, connections}</code>
Response	204 on success
Errors	INVALID_REQUEST, NOT_FOUND
Roles	Administrator, Subaccount Administrator

Request:

- `name`: name of the virtual machine on the cloud platform (a string).
- `endpointId`: unique ID (GUID) of the virtual machine (a string).
- `port`: port of the service channel for the virtual machine (a number).
- `connections`: maximal number of open connections.

Errors:

- INVALID_REQUEST (400): invalid or out of range values.
- NOT_FOUND (404): the service channel for a virtual machine with the given ID (that is, `<id>`) or the specified subaccount does not exist.

[Back to Top \[page 445\]](#)

Get all ABAP Cloud Service Channels

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ ABAPCloud</code>
Method	<code>GET</code>
Request	
Response	<pre>[{id, abapCloudTenantHost, instanceNumber, type, port, enabled, connections, state}]</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each of which represents an ABAP Cloud service channel through the following properties:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1). This identifier is unique across all types of service channels.
- `abapCloudTenantHost`: the host name (a string).
- `instanceNumber`: instance number.
- `type`: string 'ABAPCloud'.
- `port`: port of the ABAPCloud service channel (a number).
- `enabled`: boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.
- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties:
 - `connected` (a boolean flag indicating whether the channel is connected),
 - `openedConnections` (the number of open, possibly idle connections), and
 - `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).

[Back to Top \[page 445\]](#)

Get ABAP Cloud Service Channel

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ ABAPCloud/<id>
Method	GET
Request	
Response	<pre>{id, abapCloudTenantHost, instanceNumber, type, port, enabled, connections, state}</pre>
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- **id**: unique identifier for the service channel (a positive integer number, starting with 1). This identifier is unique across all types of service channels.
- **abapCloudTenantHost**: the host name (a string).
- **instanceNumber**: instance number.
- **type**: string 'ABAPCloud'.
- **port**: port of the ABAPCloud service channel (a number).
- **enabled**: boolean flag indicating whether the channel is enabled and therefore should be open.
- **connections**: maximal number of open connections.
- **state**: current connection state; this property is only available if the channel is enabled (as per property **enabled**). The value of this property is an object with the properties:
 - **connected** (a boolean flag indicating whether the channel is connected),
 - **openedConnections** (the number of open, possibly idle connections), and
 - **connectedSinceTimeStamp** (the time stamp, a UTC long number, for the first time the channel was opened/connected).

Errors:

- **NOT_FOUND (404)**: the ABAP Cloud service channel with the given ID (that is, <id>) or the specified subaccount does not exist.

Back to [Top \[page 445\]](#)

Create ABAP Cloud Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ ABAPCloud
Method	POST
Request	<pre>{abapCloudTenantHost, instanceNumber, connections}</pre>
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator, Subaccount Administrator

Request:

- `abapCloudTenantHost`: host name (a string).
- `instanceNumber`: instance number.
- `connections`: maximal number of open connections.

Errors:

- INVALID_REQUEST (400): invalid or out of range values.

Back to [Top \[page 445\]](#)

Replace ABAP Cloud Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ ABAPCloud/<id>
Method	PUT
Request	<pre>{abapCloudTenantHost, instanceNumber, connections}</pre>
Response	204 on success
Errors	INVALID_REQUEST, NOT_FOUND
Roles	Administrator, Subaccount Administrator

Request:

- `abapCloudTenantHost`: host name (a string).
- `instanceNumber`: instance number.
- `connections`: maximal number of open connections.

Errors:

- `INVALID_REQUEST` (400): invalid or out of range values.
- `NOT_FOUND` (404): the ABAP Cloud service channel with the given ID (that is, `<id>`) or the specified subaccount does not exist.

Back to [Top \[page 445\]](#)

Get All Kubernetes Cluster Service Channels

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/K8S</code>
Method	<code>GET</code>
Request	
Response	<pre>[{id, type, port, k8sCluster, k8sService, enabled, connections, state, comment}]</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

An array of objects, each of which represents a service channel for a virtual machine through the following properties:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1); this identifier is unique across all types of service channels.
- `k8sCluster`: host name to access the Kubernetes cluster (a string).
- `k8sService`: host name providing the service inside of Kubernetes cluster (a string).
- `type`: the string `'K8S'`.
- `port`: port of the service channel for the virtual machine (a number).
- `enabled`: Boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.
- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties `connected` (a Boolean flag

indicating whether the channel is connected), `openedConnections` (the number of open, possibly idle connections), and `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).

- `comment`: comment or short description; this property is not supplied if no comment was provided.

Back to [Top \[page 445\]](#)

Get Kubernetes Service Channel

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/K8S/ <id></code>
Method	<code>GET</code>
Request	
Response	<pre>[{id, type, port, k8sCluster, k8sService, enabled, connections, state, comment}]</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- `id`: unique identifier for the service channel (a positive integer number, starting with 1); this identifier is unique across all types of service channels.
- `k8sCluster`: host name to access the Kubernetes cluster (a string).
- `k8sService`: host name providing the service inside of Kubernetes cluster (a string).
- `type`: the string 'K8S'.
- `port`: port of the service channel for the virtual machine (a number).
- `enabled`: Boolean flag indicating whether the channel is enabled and therefore should be open.
- `connections`: maximal number of open connections.
- `state`: current connection state; this property is only available if the channel is enabled (as per property `enabled`). The value of this property is an object with the properties `connected` (a Boolean flag indicating whether the channel is connected), `openedConnections` (the number of open, possibly idle connections), and `connectedSinceTimeStamp` (the time stamp, a UTC long number, for the first time the channel was opened/connected).
- `comment`: comment or short description; this property is not supplied if no comment was provided.

Back to [Top \[page 445\]](#)

Create Kubernetes Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/K8S
Method	POST
Request	<pre>{k8sCluster, k8sService, port, connections, comment}</pre>
Response	201
Errors	INVALID_REQUEST
Roles	Administrator, Subaccount Administrator

Request:

- **k8sCluster**: Kubernetes cluster host name, optionally with port separated by colon (a string).
- **k8sService**: service host inside the Kubernetes cluster (a string).
- **port**: local port.
- **connections**: maximal number of open connections.
- **comment**: optional comment or short description (a string).

Errors:

- INVALID_REQUEST (400): invalid or out-of-range values.

Back to [Top \[page 445\]](#)

Replace Kubernetes Service Channel (Master Only)

URI	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/K8S/ <id>
Method	PUT
Request	<pre>{k8sCluster, k8sService, port, connections, comment}</pre>
Response	INVALID_REQUEST, NOT_FOUND

Errors

Roles	Administrator, Subaccount Administrator
-------	---

Request:

- `k8sCluster`: Kubernetes cluster host name, optionally with port separated by colon (a string).
- `k8sService`: service host inside the Kubernetes cluster (a string).
- `port`: local port.
- `connections`: maximal number of open connections.
- `comment`: optional comment or short description (a string).

Errors:

- `INVALID_REQUEST` (400): invalid or out-of-range values.
- `NOT_FOUND` (404): the Kyma service channel with the given ID (that is, `<id>`) or the specified subaccount does not exist.

Back to [Top \[page 445\]](#)

Enable/Disable Service Channel (Master Only)

Use one of the following channel types to replace `<type>` in the URI: HANA, VirtualMachine, or ABAPCloud.

⚠ Caution

The URI variant without `<type>` still works, but it is obsolete and may be removed in a future release. We recommend that you move to using `<type>` as soon as possible.

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ <type>/<id>/state</code>
Method	<code>PUT</code>
Request	<code>{enabled}</code>
Response	
Errors	<code>NOT_FOUND</code> , <code>RUNTIME_FAILURE</code>
Roles	Administrator, Subaccount Administrator

Errors:

- NOT_FOUND (404): service channel (or subaccount) does not exist.
- RUNTIME_FAILURE (500): service channel could not be opened (when attempting to set `enabled:true`).

Back to [Top \[page 445\]](#)

Delete Service Channel (Master Only)

⚠ Caution

The URI variant without `<type>` still works, but it is obsolete and may be removed in a future release. We recommend to move to using `<type>` as soon as possible.

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/ <type>/<id></code>
Method	<i>DELETE</i>
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator, Subaccount Administrator

Errors:

- NOT_FOUND (404): service channel (or subaccount) does not exist.

Back to [Top \[page 445\]](#)

Delete all Service Channels (Master Only)

ℹ 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 \[page 445\]](#)

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	<pre>[{typeDesc, details, port, enabled, connected, connectionsCount, availableConnectionsCount, connectedSinceTimeStamp}]</pre>
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:


An array of objects, each of which represents a service channel through the following properties:

- `typeDesc`: an object specifying the service channel type through the properties `typeKey` and `typeName`.
- `details`

- port
- enabled
- connected
- connectionsCount
- availableConnectionsCount
- connectedSinceTimeStamp

Back to [Top \[page 445\]](#)

Get Service Channel


Caution

Obsolete. This API is deprecated and may be removed in a future release. Use the getters for the specific service channel type (that is, HANA (database), Virtual Machine, or ABAP Cloud) which provide properties tailored for the respective channel type.

URI	/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/<id>
Method	GET
Request	
Response	[{typeDesc, details, port, enabled, connected, connectionsCount, availableConnectionsCount, connectedSinceTimeStamp}]
Errors	
Roles	Administrator, Subaccount Administrator, Display, Support

Response:

- typeDesc: an object specifying the service channel type through the properties typeKey and typeName.
- details
- port
- enabled
- connected
- connectionsCount
- availableConnectionsCount
- connectedSinceTimeStamp

Back to [Top \[page 445\]](#)

Create Service Channel

⚠ Caution

Obsolete. This API is deprecated and may be removed in a future release. Create a service channel using the API for the respective channel type, that is, HANA (database), Virtual Machine, or ABAP Cloud.

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels</code>
Method	<code>POST</code>
Request	<code>{typeKey,details,serviceNumber,connectionCount}</code>
Response	201 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request Properties:

- `typeKey`: type of service channel. Valid values are HANA_DB, HCPVM, RFC.
- `details`:
 - HANA instance name for HANA_DB
 - VM name for HCPVM
 - S/4HANA Cloud tenant host for RFC
- `serviceNumber`: service number, which is mapped to a port according to the type of service channel.
- `connectionCount`: number of connections for the channel.

Back to [Top \[page 445\]](#)

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	/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/channels/<id>
Method	PUT
Request	<pre>{typeKey,details,serviceNumber,connectionCount}</pre>
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Back to [Top \[page 445\]](#)

1.5.2.5.13 Access Control Entities

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

[Get List Of Allowed Clients \[page 465\]](#)

[Get List Of Blocked Client/User Pair \[page 466\]](#)

[Create Allowed Clients \(Master Only\) \[page 467\]](#)

[Extend Allowed Clients \(Master Only\) \[page 467\]](#)

[Create a Blocked Client/User Pair \(Master Only\) \[page 468\]](#)

[Extend a Blocked Client/User Pair \(Master Only\) \[page 468\]](#)

[Delete All Allowed Clients \(Master Only\) \[page 469\]](#)

[Delete All Blocked Client/User Pairs \(Master Only\) \[page 469\]](#)

[Remove an Item from the List of Allowed Clients \(Master Only\) \[page 470\]](#)

[Remove Items on the List of Blocked Client/User Pairs for a Given User \(Master Only\) \[page 470\]](#)

[Remove Items on the List of Blocked Client/User Pairs for a Given Client \(Master Only\) \[page 471\]](#)

[Remove an Item on the List of Blocked Client/User Pairs \(Master Only\) \[page 471\]](#)

Get List Of Allowed Clients

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

Response:

List of objects, each representing an allowed client:

- `client`: client name

Back to [Top \[page 465\]](#)

Get List Of Blocked Client/User Pair

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

Response:

List of objects, each representing a blocked client and user:

- `client`: client name
`user`: user name

[Back to Top \[page 465\]](#)

Create Allowed Clients (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ allowedClients</code>
Method	<i>POST</i>
Request	<code>[client]</code>
Response	201 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request:

List of strings, each representing an allowed client.

[Back to Top \[page 465\]](#)

Extend Allowed Clients (Master Only)

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

Request:

List of strings, each representing an allowed client.

Back to [Top \[page 465\]](#)

Create a Blocked Client/User Pair (Master Only)

URI	<code>/api/v1/configuration/subaccounts/ <regionHost>/<subaccount>/systemMapping/ <virtualHost>:<virtualPort>/ blockedClientUsers</code>
Method	<i>POST</i>
Request	<code>[{client, user}]</code>
Response	201 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request:

List of objects, each representing a blocked client and user:

- `client`: ABAP client
- `user`: ABAP user

Back to [Top \[page 465\]](#)

Extend a Blocked Client/User Pair (Master Only)

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

Request	<code>[{client, user}]</code>
Response	204 on success
Errors	
Roles	Administrator, Subaccount Administrator

Request:

List of objects, each representing a blocked client and user:

- `client`: ABAP client
- `user`: ABAP user

Back to [Top \[page 465\]](#)

Delete All Allowed Clients (Master Only)

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

Back to [Top \[page 465\]](#)

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 \[page 465\]](#)

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 \[page 465\]](#)

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

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

Back to [Top \[page 465\]](#)

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

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

Back to [Top \[page 465\]](#)

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

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

Back to [Top \[page 465\]](#)

1.5.2.5.14 Examples

Find examples on how to use the Cloud Connector's configuration REST APIs.

Concept

The sample code in this section ([download](#) zip file) demonstrates how to use the REST APIs provided by Cloud Connector to perform various configuration tasks.

Starting with a freshly installed Cloud Connector, the samples include initial configuration of the Cloud Connector instance, connectivity setup, high availability configuration, and common tasks like backup/restore operations, as well as integration with solution management.

The examples are implemented in *Kotlin*, a simple Java VM-based language. However, even if you are using a different language, they still show the basic use of the APIs and their parameters for specific configuration purposes.

If you are not familiar with Kotlin, find a brief introduction and some typical statements below.

[REST API Parameters \[page 473\]](#)

[REST API Invocation \[page 473\]](#)

[How To Use the Examples \[page 474\]](#)

REST API Parameters

In almost all requests and responses, structures are encoded in JSON format. To describe the parameter details, we use Kotlin data classes.

This class represents a structure that you can use as value in a request or response:

```
data class OnlyPropertiesNamesAreRelevant(  
    val user: String,  
    val password: String  
)
```

The JSON representation for that class is

```
{"user":<userValue>, "password":<passwordValue>}
```

Encoded in Kotlin this string looks like

```
"""{"user":"$userValue", "password":"$passwordValue"}"""
```

or as an object:

```
val credentials = OnlyPropertiesNamesAreRelevant(userValue, passwordValue)
```

Back to [Concept \[page 472\]](#)

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) {  
(f)        is Result.Failure -> processRequestError(result.error)  
            is Result.Success -> println("returned: ${result.get()}")  
        }  
    }  
    .join()
```

- (a) - *Fuel* is an HTTP framework used in the examples. Important is the verb after *Fuel* - it is the REST API method.
- (b) - Adds a request header `Connection: close`, which forces a connection close after request. In the examples, this header is defined on *FuelManager* for all calls.

- (c) - Basic authentication is used for the call with user and password.
- (d) - HTTP requests with parameters require mostly JSON. The `jsonBody`-method adds the header `Content-Type: application/json` and serializes the provided object credentials to JSON. Requests without body like [DELETE](#) or [GET](#) omit body methods.
- (e) - The `responseObject<ClassType>`-method adds the header `Accept: application/json` and de-serializes the response to the specified class type. Some APIs do not have any response or non-JSON response. In such cases, the method `response` is used instead of `responseObject<ClassType>`.
- (f) - The way, how Kotlin decides between success (2xx) and failed responses. For details on the possible response status, see [Configuration REST APIs \[page 367\]](#).

Back to [Concept \[page 472\]](#)

How To Use the Examples

Some common details, used by different examples, were extracted to the `scenario.json` configuration file. This lets you use meaningful names like `config.master!!user` in the examples. The file is loaded by

```
val config = loadScenarioConfiguration()
```

Each example also invokes

```
disableTrustChecks()
```

This method disables all SSL-related checks.

⚠ Caution

`disableTrustChecks()` is only used for test purposes. *Do not* use it in a productive environment.

Both methods, as well as some common REST API parameter structures (data classes) are defined in the file [Scenario Configuration \[page 476\]](#). This is the only help class under `sources/`.

When using the examples, start with [Initial Configuration \[page 477\]](#).

Prerequisite is a freshly installed Cloud Connector.

After a mandatory password change and defining the high availability role of the Cloud Connector instance (master or shadow), the example demonstrates how to provide a description for the instance and how to set up the UI and system certificates.

Once the initial configuration is done, you can optionally proceed with these steps:

- Connect to your subaccount on BTP ([Subaccount Configuration \[page 479\]](#))
- Create or restore a configuration backup ([Backup And Restore Configuration \[page 486\]](#))
- Configure and connect high availability instances ([High Availability Settings \[page 484\]](#))
- Integrate the Cloud Connector with the solution management infrastructure ([Solution Management Integration \[page 487\]](#))

[Back to Concept \[page 472\]](#)

Related Information

[scenario.json \[page 475\]](#)

[Source Files \[page 475\]](#)

1.5.2.5.14.1 scenario.json

Sample Code

```
{
  "subaccount": {
    "regionHost": "cf.eu10.hana.ondemand.com",
    "subaccount": "11aabbcc-7821-448b-9ecf-a7d986effa7c",
    "user": "xxx",
    "password": "xxx"
  },
  "master": {
    "url": "https://localhost:8443",
    "user": "Administrator",
    "password": "test"
  },
  "shadow": {
    "url": "https://localhost:8444",
    "user": "Administrator",
    "password": "test"
  }
}
```

1.5.2.5.14.2 Source Files

[Scenario Configuration \[page 476\]](#)

[Initial Configuration \[page 477\]](#)

[Subaccount Configuration \[page 479\]](#)

[High Availability Settings \[page 484\]](#)

[Backup And Restore Configuration \[page 486\]](#)

[Solution Management Integration \[page 487\]](#)

1.5.2.5.14.2.1 Scenario Configuration

Sample Code

```
package com.sap.scc.examples
import com.github.kittinunf.fuel.core.FuelError
import com.google.gson.Gson
import java.io.File
import java.security.SecureRandom
import java.security.cert.X509Certificate
import javax.net.ssl.*
data class CloudConnector(
    val url: String,
    var user: String,
    var password: String
)
fun disableTrustChecks() {
    try {
        HttpURLConnection.setDefaultHostnameVerifier { hostname: String,
session: SSLSession -> true }
        val context: SSLContext = SSLContext.getInstance("TLS")
        val trustAll: X509TrustManager = object : X509TrustManager {
            override fun checkClientTrusted(chain: Array<X509Certificate>,
authType: String) {}
            override fun checkServerTrusted(chain: Array<X509Certificate>,
authType: String) {}
            override fun getAcceptedIssuers(): Array<X509Certificate> {
                return arrayOf()
            }
        }
        context.init(null, arrayOf(trustAll), SecureRandom())
        HttpURLConnection.setDefaultSSLSocketFactory(context.socketFactory)
    } catch (e: Exception) {
        e.printStackTrace()
    }
}
class ScenarioConfiguration {
    var subaccount: SubaccountParameters? = null
    var master: CloudConnector? = null
    var shadow: CloudConnector? = null
}
data class SubaccountParameters(
    val regionHost: String,
    val subaccount: String,
    val user: String,
    val password: String,
    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. ")
}
```

1.5.2.5.14.2.2 Initial Configuration

Sample Code

```
package com.sap.scc.examples
//import com.sap.scc.examples.SccCertificate
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.BlobDataPart
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.Method
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
import java.io.ByteArrayInputStream
import java.io.File
/*
 * This example shows how to use REST APIs to perform the initial configuration
 * of a master instance
 * after installing and starting the Cloud Connector.
 * As a prerequisite you need to install and start the Cloud Connector.
 * The example begins with changing the initial password, setting the instance
 * to the master role, edit the description,
 * and upload UI and system certificates.
 * For the certificates used by Cloud Connector in order to access the UI and
 * for the system certificate used to access backend systems,
 * we simply upload the already available PKCS#12 certificates uiCert.p12 and
 * systemCert.p12 encrypted with the password "test1234".
 * Cloud Connector also provides other options for certificate management,
 * please take a look at the documentation.
 * The configuration details for master and shadow instances can be found in
 * scenario.json.
 */
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
    certificate.
    //So for this demonstration use case we need to deactivate all trust
    checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
    efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
    //Load configuration from property file
    val config = loadScenarioConfiguration()
    //Change the initial password
    Fuel.put("${config.master!!.url}/api/v1/configuration/connector/
    authentication/basic")
        .authentication().basic(config.master!!.user, "manage")
        .body("""{"oldPassword":"manage", "newPassword":"$
    {config.master!!.password}""")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
            }
        }
    }
```

```

        is Result.Success -> println("Password successfully set to $
{config.master!!.password}")
    }
    }
    .join()
    //Set the High-Availability Role to master, use "shadow" if you want to
    set it to shadow
    Fuel.put("${config.master!!.url}/api/v1/configuration/connector/haRole")
        .authentication().basic(config.master!!.user,
    config.master!!.password)
        .body("master")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("high-availability role
successfully set to 'master'")
            }
        }
    .join()
    //Edit Common Description
    Fuel.put("${config.master!!.url}/api/v1/configuration/connector")
        .authentication().basic(config.master!!.user,
    config.master!!.password)
        .body("{\"description\":\"<description of Cloud Connector
instance>\"}")
        .responseObject<SccDescriptionResponse> { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println(result.get())
            }
        }
    .join()
    //Upload a PKCS#12 Certificate as UI Certificate
    val uiCertificateFormData = listOf("password" to "test1234",
    "keyPassword" to "test1234")
    Fuel.upload(
        "${config.master!!.url}/api/v1/configuration/connector/ui/
    uiCertificate",
        Method.PUT,
        uiCertificateFormData
    )
        .add(BlobDataPart(ByteArrayInputStream(File("uiCert.p12").readBytes()),
    , name = "pkcs12"))
        .authentication().basic(config.master!!.user,
    config.master!!.password)
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("PKCS#12 Certificate
'uiCert.p12' successfully uploaded")
            }
        }
    .join()
    //Upload a PKCS#12 Certificate as as System Certificate
    val systemCertificateFormData = listOf("password" to "test1234",
    "keyPassword" to "test1234")
    Fuel.upload(
        "${config.master!!.url}/api/v1/configuration/connector/onPremise/
    systemCertificate",
        Method.PUT,
        systemCertificateFormData
    )
        .add(BlobDataPart(ByteArrayInputStream(File("systemCert.p12").readByte
s()), name = "pkcs12"))
        .authentication().basic(config.master!!.user,
    config.master!!.password)
        .response { _, _, result ->
            when (result) {

```

```

        is Result.Failure -> processRequestError(result.error)
        is Result.Success -> println("PKCS#12 Certificate
'systemCert.p12' successfully uploaded")
    }
    }
    .join()
}
//Data structures used by REST calls in this scenario
data class SccDescriptionResponse(
    var role: String,
    var description: String
)

```

1.5.2.5.14.2.3 Subaccount Configuration

Sample Code

```

package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.jsonBody
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
/*
 * This example shows how to use REST APIs to configure and connect a
 * subaccount in cloud connector.
 * The example begins with (1) connecting of the subaccount, then we create a
 * system (2) for an HTTP service
 * and (3) for an RFC service.
 *
 * The configuration details for master and shadow instances can be found in
 * scenario.json.
 */
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
    certificate.
    //So for this demonstration use case we need to deactivate all trust
    checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
    efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
    //1.1. Create and connect subaccount
    //Load configuration from property file
    val config = loadScenarioConfiguration()
    //Some cloud regions require 2-Factor-Authentication
    println("Enter MFA (aka 2FA) token, if required: ")
    var token = readLine() ?: ""
    //Parameters required to establish the connection to the subaccount (aka
    the secure tunnel)
    var subaccountCreateData = SubaccountConfiguration(
        config.subaccount!!.regionHost, config.subaccount!!.subaccount,
        config.subaccount!!.user, "" + config.subaccount!!.password + token,
        locationId = config.subaccount!!.locationId
    )
    //Optional: Initialize the map to work with generated _links

```

```

//If you don't like to use _links, you can easily compute the entity links
var subaccountLinks: Map<String, HalLink> = mapOf()
Fuel.post("${config.master!!.url}/api/v1/configuration/subaccounts")
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .jsonBody(subaccountCreateData)
    .responseObject<SubaccountInfo> { _, response, result ->
        when (result) {
            is Result.Success -> {
                val subaccountLocation =
response.header("Location").first()
                println("1.1. subaccount was created under:
$subaccountLocation")
                println("subaccount: ${result.get()} ")
                //GET details as SubaccountInfo every time possible by
invoking
                //https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>
                subaccountLinks = result.get()._links
            }
            is Result.Failure -> processRequestError(result.error)
        }
    }
    .join()
//1.2. From time to time it is necessary to extend the validity of the
subaccount - refresh subaccount
//Again some cloud landscapes require 2-Factor-Authentication
println("Enter MFA (aka 2FA) token for refresh, if required: ")
token = readLine() ?: ""
Fuel.post(subaccountLinks["validity"]!!.href)
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .jsonBody(SubaccountRefreshCred(config.subaccount!!.user, "" +
config.subaccount!!.password + token))
    .responseObject<SubaccountInfo> { _, _, result ->
        when (result) {
            is Result.Success -> println("1.2. validity of the subaccount
was extended")
            is Result.Failure -> processRequestError(result.error)
        }
    }
    .join()
//2.1. Create a system mapping (aka access control) for an HTTP service
val httpSystem = SystemMapping(
    "virtual.host", "vport", "local", "lport",
CommunicationProtocol.HTTPS,
BackendType.abapSys, hostInHeader = HostInHeader.INTERNAL
)
var httpSystemLinks: Map<String, HalLink> = mapOf()
Fuel.post(subaccountLinks["systemMappings"]!!.href)
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .jsonBody(httpSystem)
    .responseString { _, response, result ->
        when (result) {
            is Result.Success -> {
                val httpSystemMappingLocation =
response.header("Location").first()
                println("2.1. system mapping was created under:
$httpSystemMappingLocation")
                //GET the details about the system mapping by invoking
                //https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>
                Fuel.get(httpSystemMappingLocation)
                    .authentication().basic(config.master!!.user,
config.master!!.password)
                    .responseObject<SystemMapping> { _, _, result ->
                        when (result) {

```

```

                is Result.Success -> {
                    println("system mapping: ${result.get()}")
                    httpSystemLinks = result.get()._links
                }
                is Result.Failure ->
processRequestError(result.error)
            }
        }
        .join()
    }
    is Result.Failure -> processRequestError(result.error)
}
}
.join()
//2.2 Add an allowed resource to the system mapping. Since this system
mapping points to an HTTP service, use HTTP resource parameters
Fuel.post(httpSystemLinks["resources"]!!!.href)
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .jsonBody(HttpResource("/", exactMatchOnly = false))
    .responseString { _, response, result ->
        when (result) {
            is Result.Success -> {
                val httpResourceLocation =
response.header("Location").first()
                println("2.2. http resource was created under:
$httpResourceLocation")
                //GET the details about the resource by invoking
                //https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>/<encoded-id>
                //<encoded-id> -> replace '/' with '-'
                Fuel.get(httpResourceLocation)
                    .authentication().basic(config.master!!.user,
config.master!!.password)
                    .responseObject<HttpResource> { _, _, result ->
                        when (result) {
                            is Result.Success -> println("http resource:
${result.get()}")
                            is Result.Failure ->
processRequestError(result.error)
                        }
                    }
                .join()
            }
            is Result.Failure -> processRequestError(result.error)
        }
    }
}
.join()
//3.1 Create a system mapping for an RFC service
var rfcSystemLinks: Map<String, HalLink> = mapOf()
Fuel.post(subaccountLinks["systemMappings"]!!!.href)
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .jsonBody(
        SystemMapping(
            "virtual.host",
            "rfcport",
            "local",
            "rfcport",
            CommunicationProtocol.RFCS,
            BackendType.abapSys
        )
    )
    .responseString { _, response, result ->
        when (result) {
            is Result.Success -> {
                val rfcSystemMappingLocation =
response.header("Location").first()

```

```

        println("3.1. system mapping was created under:
$rfcSystemMappingLocation")
        //GET the details about the system mapping by invoking
        //https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>
        Fuel.get(rfcSystemMappingLocation)
            .authentication().basic(config.master!!.user,
config.master!!.password)
            .responseObject<SystemMapping> { _, _, result ->
                when (result) {
                    is Result.Success -> {
                        println("system mapping: ${result.get()}")
                        rfcSystemLinks = result.get()._links
                    }
                    is Result.Failure ->
processRequestError(result.error)
                }
            }
        }
        .join()
    }
    is Result.Failure -> processRequestError(result.error)
}
}
.join()
//3.2 Now add the function module RFC_SYSTEM_INFO as an allowed resource
to the system mapping
val rfcResource = RfcResource("RFC_SYSTEM_INFO")
Fuel.post(rfcSystemLinks["resources"]!!.href)
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .jsonBody(rfcResource)
    .responseString { _, response, result ->
        when (result) {
            is Result.Success -> {
                val resourceLocation = response.header("Location").first()
                println("3.2. rfc resource was created under:
$resourceLocation")
                //GET the details about the resource by invoking
                //https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>/<encoded-id>
                //<encoded-id> -> replace '/' with '-'
                Fuel.get(resourceLocation)
                    .authentication().basic(config.master!!.user,
config.master!!.password)
                    .responseObject<RfcResource> { _, _, result ->
                        when (result) {
                            is Result.Success -> println("rfc resource: $
{result.get()}")
                            is Result.Failure ->
processRequestError(result.error)
                        }
                    }
                }
            }
        }
        .join()
    }
    is Result.Failure -> processRequestError(result.error)
}
}
.join()
}
}
}
//Data structures used by REST calls in this scenario
//Nullable properties are optional
data class SubaccountConfiguration(
    var regionHost: String,
    var subaccount: String,
    var cloudUser: String,
    var cloudPassword: String,
    var displayName: String? = null,
    var locationId: String? = null

```

```

)
data class SubaccountInfo(
    val displayName: String,
    val regionHost: String,
    val subaccount: String,
    val tunnel: SubaccountTunnelInfo,
    val user: String,
    val _links: Map<String, HalLink>
)
data class SubaccountTunnelInfo(
    val state: String, //Connected
    val connectedSince: String, //timestamp formatted with client locale
    val connections: Int,
    val applicationConnections: List<String>,
    val serviceChannels: List<String>,
    val subaccountCertificate: X509CertificateInfo,
)
data class X509CertificateInfo(
    val notAfter: String,
    val notBefore: String,
    val subjectDN: String,
    val issuer: String
)
data class HalLink(val href: String)
data class SubaccountRefreshCred(
    val cloudUser: String,
    val cloudPassword: String
)
data class SystemMapping(
    val virtualHost: String,
    val virtualPort: String,
    val localhost: String,
    val localPort: String,
    val protocol: CommunicationProtocol,
    val backendType: BackendType,
    val sncPartnerName: String? = null,
    val hostInHeader: HostInHeader? = null,
    val sapRouter: String? = null,
    val authenticationMode: AuthenticationMode = AuthenticationMode.NONE,
    val description: String = "",
) {
    val _links: Map<String, HalLink> = mapOf()
}
enum class CommunicationProtocol {
    HTTP, HTTPS, RFC, RFCS, LDAP, LDAPS, TCP, TCPS
}
enum class BackendType {
    abapSys, netweaverCE, applServerJava, BC, PI, hana, netweaverGW,
    otherSAPsys, nonSAPsys
}
enum class HostInHeader {
    INTERNAL, VIRTUAL
}
enum class AuthenticationMode {
    NONE, X509_CERTIFICATE, X509_CERTIFICATE_LOCAL, KERBEROS
}
data class HttpResource(
    val id: String, //URI path
    val enabled: Boolean = true,
    val exactMatchOnly: Boolean = true,
    val websocketUpgradeAllowed: Boolean = true,
    val description: String = ""
)
data class RfcResource(
    val id: String, //Function module or prefix for function module
    val enabled: Boolean = true,
    val exactMatchOnly: Boolean = true,
    val description: String = ""
)

```

)

1.5.2.5.14.2.4 High Availability Settings

Sample Code

```
package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.Headers
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.jsonBody
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
import java.net.URL
/*
 * This example shows how to use REST APIs to change high availability settings
 * of the Cloud Connector instances.
 * As a prerequisite you need to install and start a shadow and a master
 * instance.
 * Afterwards perform the initial configuration of the master instance (see
 * InitialConfiguration.kt).
 *
 * The configuration details for master and shadow instances can be found in
 * scenario.json.
 */
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
    //certificate.
    //So for this demonstration use case we need to deactivate all trust
    //checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
    //efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
    //Load configuration from property file
    val config = loadScenarioConfiguration()
    //The high-availability role 'master' in Cloud Connector instance
    //is already set in the initial configuration.
    //Set the high-availability role to 'shadow' in Cloud Connector instance
    'shadow'
    Fuel.put("${config.shadow!!.url}/api/v1/configuration/connector/haRole")
        .authentication().basic(config.shadow!!.user,
        config.shadow!!.password)
        .body("shadow")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("high-availability role
                successfully set to 'shadow'")
            }
        }
        .join()
    //Enable HA in the master instance and define the allowed shadow hosts
    Fuel.put("${config.master!!.url}/api/v1/configuration/connector/ha/master/
    config")
        .header(Headers.CONTENT_TYPE, "application/json")
}
```



```

        .authentication().basic(config.master!!.user,
config.master!!.password)
        .body("""{"haEnabled":"true", "allowedShadowHost": "$
{URL(config.shadow!!.url).host}""")
        .responseObject<HAMasterConfiguration> { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> {
                    val masterConfig = result.get()
                    println("Master configuration: $masterConfig ")
                }
            }
        }
    }.join()
    //The configuration of the shadow instance
    var shadowConfig: HAShadowConfiguration? = null
    //Get the current configuration
    Fuel.get("${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": "${config.master!!.password}""")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("Master (${config.master!!.url})
and shadow (${config.shadow!!.url}) successfully connected")
            }
        }
    }.join()
}
//Data structures used by REST calls in this scenario
//Structure used to read the high availability settings for a Cloud
Connector master instance
data class HAMasterConfiguration(
    var haEnabled: Boolean? = null,
    var allowedShadowHost: String? = null
)

```

```
//Structure used to read and edit the high availability settings for a Cloud
Connector shadow instance
data class HAsShadowConfiguration(
    var masterPort: String,
    var masterHost: String,
    var ownHost: String? = null,
    var checkIntervalInSeconds: Int?,
    var takeoverDelayInSeconds: Int?,
    var connectTimeoutInMillis: Int?,
    var requestTimeoutInMillis: Int?
)
```

1.5.2.5.14.2.5 Backup And Restore Configuration

Sample Code

```
package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.BlobDataPart
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.Headers
import com.github.kittinunf.fuel.core.Method
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.result.Result
import java.io.ByteArrayInputStream
import java.io.File
/*
This example shows how to use REST APIs to perform the Backup and Restore of
the Cloud Connector configuration.
As a prerequisite you have a stable Cloud Connector configuration and you
want to save the backup for later use.

The configuration details can be found in scenario.json.
*/
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
    certificate.
    //So for this demonstration use case we need to deactivate all trust
    checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
    efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
    //Load configuration from property file
    val config = loadScenarioConfiguration()
    //Create the backup configuration of the cloud connector "master".
    Fuel.post("${config.master!!.url}/api/v1/configuration/backup")
        .header(Headers.CONTENT_TYPE, "application/json")
        .authentication().basic(config.master!!.user,
        config.master!!.password)
        .body("""{"password":"my-very-secret-password"}""")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success ->
                    File("backup.zip").writeBytes(result.get())
            }
        }
```

```

    }
    .join()
    //Restore the backup configuration if some fatal accidental changes
    should be reverted
    val formData = listOf("password" to "my-very-secret-password")
    Fuel.upload("${config.master!!.url}/api/v1/configuration/backup",
    Method.PUT, formData)
        .add(BlobDataPart(ByteArrayInputStream(File("backup.zip").readBytes()),
    , name = "backup"))
        .authentication().basic(config.master!!.user,
    config.master!!.password)
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("Backup configuration
    successfully restored in (${config.master!!.url}) ")
            }
        }
    .join()
}

```

1.5.2.5.14.2.6 Solution Management Integration

🔗 Sample Code

```

package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.jsonBody
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
import java.io.File
/*
    This example shows how to use REST APIs to configure the integration with
    SAP solution management.
    In most cases it should be only necessary to pass a boolean flag in order to
    enable or
    disable solution management integration. It is expected that SAP host agent
    is already
    installed on the host as prerequisite for solution management integration.

    This example executes requests against master and shadow instances.
    The shadow instance is optional. Ignore the requests to shadow instance if
    there is only
    a master instance in your environment.

    The configuration details for master and shadow instances can be found in
    scenario.json.
    */
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
    certificate.
    //So for this demonstration use case we need to deactivate all trust
    checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
    efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")

```

```

//Add output of cURL commands for revision
//FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
//Load configuration from property file
val config = loadScenarioConfiguration()
//First we check the current configuration of solution management
Fuel.get("${config.master!!.url}/api/v1/configuration/connector/
solutionManagement")
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .responseObject<SolutionManagementConfiguration> { _, _, result ->
        when (result) {
            is Result.Success -> println("response: ${result.get()}")
            is Result.Failure -> processRequestError(result.error)
        }
    }
    .join()
//Configure the hostAgent path on the shadow instance
//Invoking this API on a shadow instance changes only the configuration.
//Only when invoking it on a master instance it also activates solution
management integration.
//Note: This step is not required if host agent is installed at the
default location
Fuel.post("${config.shadow!!.url}/api/v1/configuration/connector/
solutionManagement")
    .authentication().basic(config.shadow!!.user,
config.shadow!!.password)
    .jsonBody(SolutionManagementConfiguration(hostAgentPath = "/path/to/
hostAgent"))
    .response { _, _, result ->
        when (result) {
            is Result.Success -> println("new path to host agent was set")
            is Result.Failure -> processRequestError(result.error)
        }
    }
    .join()
//Turns on the solution management integration.
//Note: this operation is possible only on the master instance.
//In case of an HA setup, the flag enabled is propagated to the shadow
automatically.
//Note: optionally you can set here the new path for host agent and
enable DSR.
Fuel.post("${config.master!!.url}/api/v1/configuration/connector/
solutionManagement")
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .jsonBody(SolutionManagementConfiguration())
    .response { _, _, result ->
        when (result) {
            is Result.Success -> println("solution management is
activated")
            is Result.Failure -> processRequestError(result.error)
        }
    }
    .join()
//Turns off the solution management integration.
//Note: this operation is possible only on the master instance.
//In case of an HA setup, the flag not enabled is propagated to the
shadow automatically
Fuel.delete("${config.master!!.url}/api/v1/configuration/connector/
solutionManagement")
    .authentication().basic(config.master!!.user,
config.master!!.password)
    .response { _, _, result ->
        when (result) {
            is Result.Success -> println("solution management is
deactivated")
            is Result.Failure -> processRequestError(result.error)
        }
    }

```

```

    }
    .join()
    //Registration file with LMDB model can be downloaded by following
    request.
    //This file can be used for a manual upload to solution management.
    //If solution management integration is active,the updates are triggered
    by configuration changes automatically.
    Fuel.get("${config.master!!.url}/api/v1/configuration/connector/
    solutionManagement/registrationFile")
        .authentication().basic(config.master!!.user,
    config.master!!.password)
        .response { _, _, result ->
            when (result) {
                is Result.Success ->
    File("lmdbModel.zip").writeBytes(result.get())
                is Result.Failure -> processRequestError(result.error)
            }
        }
    .join()
}
//Data structures used by REST calls in this scenario
//Nullable properties are optional
data class SolutionManagementConfiguration(
    var hostAgentPath: String? = null,
    var enabled: Boolean? = null,
    var dsrEnabled: Boolean? = null
)

```

1.5.2.6 Configure an On-Premise User Store

Configure SAP BTP Java applications to use your corporate LDAP server or on-premise SAP system as a user store.

Prerequisites

- You have configured your Java cloud application to use an on-premise user provider and to consume its users via the Cloud Connector. To do this, execute the following command:

```

neo deploy --host <host> --account <subaccount name> --application
<application name> --source <path to WAR file> --user <e-mail or user name>
--vm-arguments "-Dcom.sap.cloud.security.um.user_provider_name=onpremise
-Dcom.sap.cloud.security.um.destination_name=onpremiseconnector"

```

- You have created a connectivity destination to configure the on-premise user provider, using the following parameters:

```

Name=onpremiseconnector
Type=HTTP
URL= http://scc.scim:80/scim/v1
Authentication=NoAuthentication
CloudConnectorVersion=2
ProxyType=OnPremise

```

- You are using only one domain for user authentication. Authentication to multiple domains including sub-domains is not supported.

For more information, see also [On-Premise User Store](#).

Context

If you configure your SAP BTP applications to use the corporate LDAP server or on-premise SAP system as a user store, the platform doesn't need to keep the entire user database but requests the necessary information from the on-premise user store. Java applications running on SAP BTP can use the on-premise system to check credentials, search for users, and retrieve details. In addition to the user information, the cloud application may request information about the groups a user belongs to.

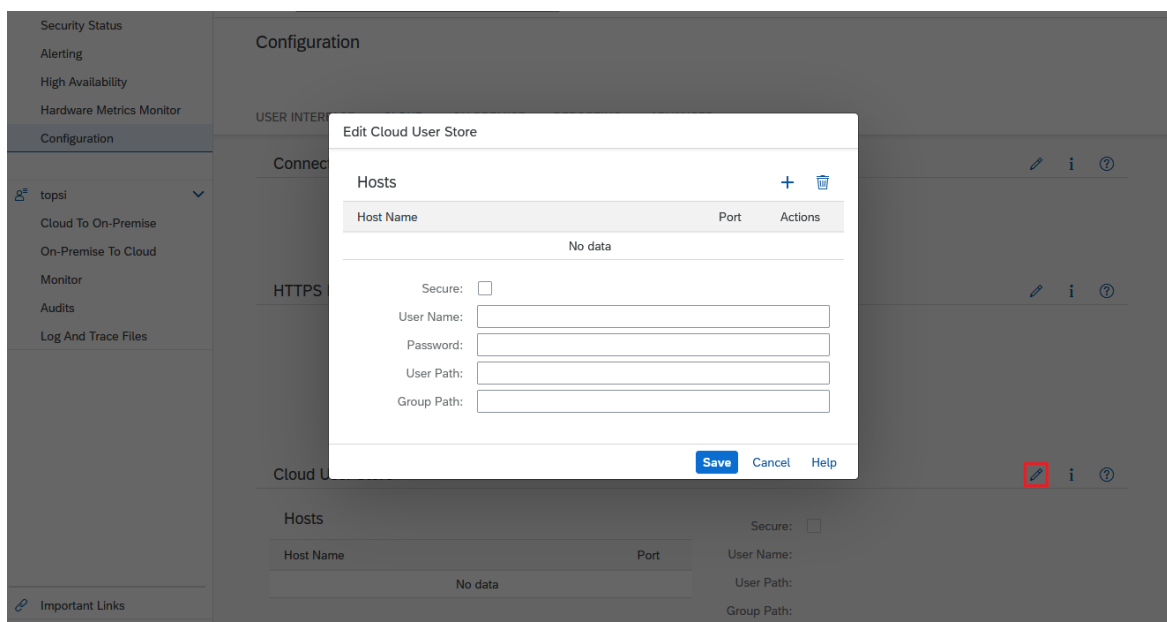
One way a Java cloud application can define user authorizations is by checking a user's membership to specific groups in the on-premise user store. The application uses the roles for the groups defined in SAP BTP. For more information, see [Managing Roles](#).

Note

The configuration steps below are applicable only for Microsoft Active Directory (AD).

Procedure

1. From the main menu, choose [Configuration](#).
2. From the [Cloud User Store](#) section on the [Cloud](#) tab, choose [Edit](#).



3. To connect to LDAP using TLS, select [Secure](#).
4. Manage the hosts and ports for your LDAP servers:

- Choose the [Add](#) icon to add a host.

Note

Multiple hosts are currently not supported.

- Choose [Edit](#) to edit the selected host.
 - Choose [Delete](#) to delete the selected hosts.
5. Enter the user name and password for the service user that is used to contact the LDAP system.

Note

The user name must be fully qualified, including the AD domain suffix, for example, [john.smith@mycompany.com](#).

6. In [User Path](#), specify the LDAP subtree that contains the users.
7. In [Group Path](#), specify the LDAP subtree that contains the groups.
8. Choose **Save**.

Related Information

[Using an SAP System as an On-Premise User Store](#)

[Use LDAP for User Administration \[page 521\]](#)

[Managing Destinations](#)

1.5.2.7 Using Service Channels

Configure Cloud Connector service channels to connect your on-premise network to specific services on SAP BTP or to S/4HANA Cloud.

Context

Cloud Connector service channels provide access from an external network to certain services on SAP BTP, or to S/4HANA Cloud. 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 SAP HANA Database 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.
Virtual Machine on SAP BTP	You can use the virtual machine (VM) service channel to access a SAP BTP VM using an SSH client, and adjust it to your needs.
RFC Connection to SAP BTP ABAP environment and S/4HANA Cloud	The service channel for RFC supports calls from on-premise systems to the SAP BTP ABAP environment and S/4HANA Cloud using RFC.

Next Steps

[Configure a Service Channel for an SAP HANA Database \[page 492\]](#)

[Connect DB Tools to SAP HANA via Service Channels \[page 495\]](#)

[Configure a Service Channel for Virtual Machine \[page 496\]](#)

[Configure a Service Channel for RFC \[page 498\]](#)

Related Information

[Service Channels: Port Overview \[page 500\]](#)

1.5.2.7.1 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.

⚠ Restriction

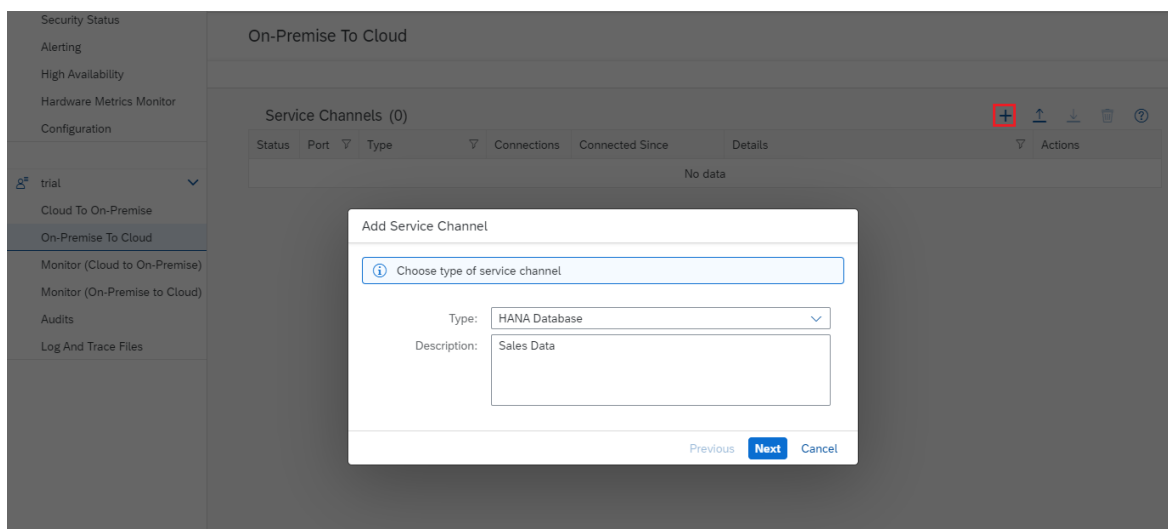
This feature is only available for the Neo environment.

📘 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

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



3. In the *Add Service Channel* dialog, leave the default value **HANA Database** in the *<Type>* field.
4. Optionally, provide a *Description* that explains what the HANA DB service channel is used for.
5. Choose *Next*.
6. 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.

📘 Note

The SAP HANA instance name is case-sensitive.

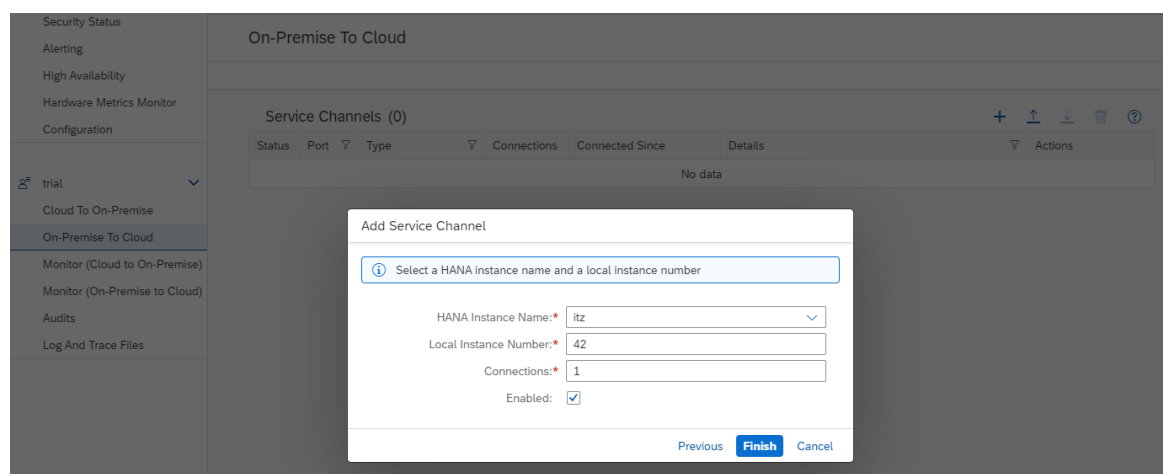
In the **Cloud Foundry** environment, the format of the SAP HANA instance name includes the Cloud Foundry space name, the database name and the database ID.

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<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

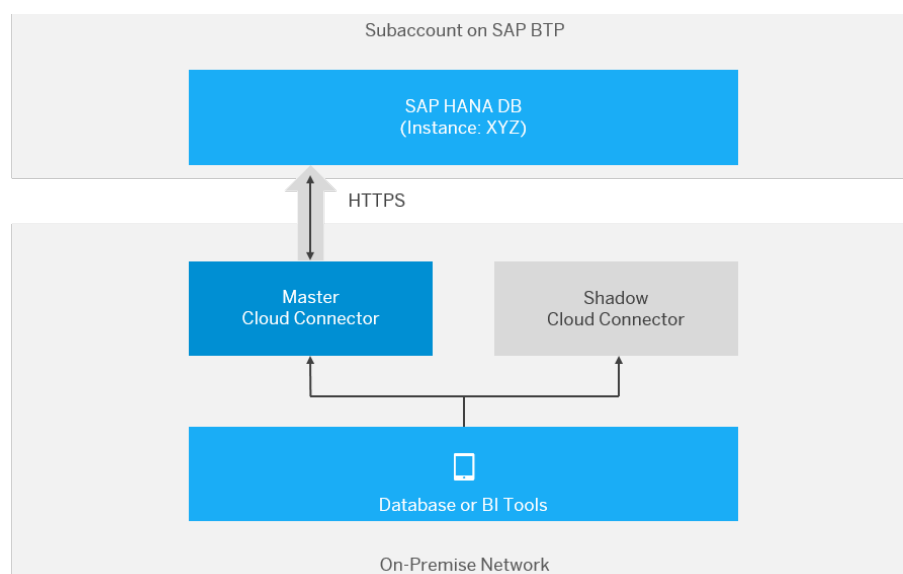
Once you have established an SAP HANA Database service channel, you can connect on-premise database or BI tools to the selected SAP HANA database in the cloud. This can be done by using `<cloud_connector_host>:<local_HANA_port>` in the JDBC/ODBC connect strings.

See [Connect DB Tools to SAP HANA via Service Channels \[page 495\]](#).

1.5.2.7.2 Connect DB Tools to SAP HANA via Service Channels

Context

You can connect database, BI, or replication tools running in on-premise network to an SAP HANA database on SAP BTP using service channels of the Cloud Connector. You can also use the high availability support of the Cloud Connector on a database connection. The picture below shows the landscape in such a scenario.



Follow the steps below to set up failover support, configure a service channel, and connect on-premise DB tools via JDBC or ODBC to the SAP HANA database.

- For more information on using SAP HANA instances, see [Using an SAP HANA XS Database System](#)
- For the connection string via ODBC you need a corresponding database user and password (see step 4 below). See also: [Creating Database Users](#).
- Find detailed information on failover support in the *SAP HANA Administration Guide*: [Configuring Clients for Failover](#).

Note

This link points to the latest release of *SAP HANA Administration Guide*. Refer to the [SAP BTP Release Notes](#) to find out which SAP HANA SPS is supported by SAP BTP. Find the list of guides for earlier releases in the *Related Links* section below.

Procedure

1. To establish a highly available connection to one or multiple SAP HANA instances in the cloud, we recommend that you make use of the failover support of the Cloud Connector. Set up a master and a shadow instance. See [Install a Failover Instance for High Availability \[page 538\]](#).
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 \[page 492\]](#).
3. Connect on-premise DB tools via JDBC to the SAP HANA database by using the following connection string:

Example:

```
jdbc:sap://<cloud-connector-master-host>:30115;<cloud-connector-shadow-host>:30115[/?<options>]
```

The SAP HANA JDBC driver supports failover out of the box. All you need is to configure the shadow instance of the Cloud Connector as a failover server in the JDBC connection string. The different options supported in the JDBC connection string are described in: [Connect to SAP HANA via JDBC](#)

4. You can also connect on-premise DB tools via ODBC to the SAP HANA database. Use the following connection string:

```
"DRIVER=HDBODBC32;UID=<user>;PWD=<password>;SERVERNODE=<cloud-connector-master-host>:30115,<cloud-connector-shadow-host>:30115;"
```

Related Information

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

1.5.2.7.3 Configure a Service Channel for Virtual Machine

Context

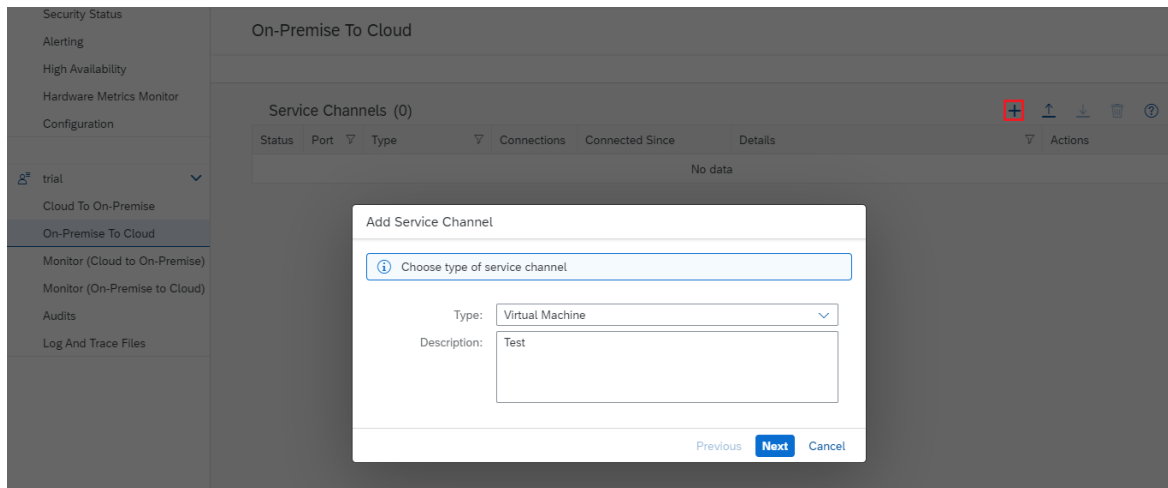
You can establish a connection to a virtual machine (VM) in the SAP BTP that is not directly exposed to external access. Use ► [On-Premise to Cloud](#) ► [Service Channels](#) ► in the Cloud Connector. You can use the service channel to manage the VM and adjust it to your needs.

📘 Note

The following procedure requires that you have created a VM in your subaccount.

Procedure

1. From your subaccount menu, choose *On Premise To Cloud*.
2. Choose the *Add* icon.

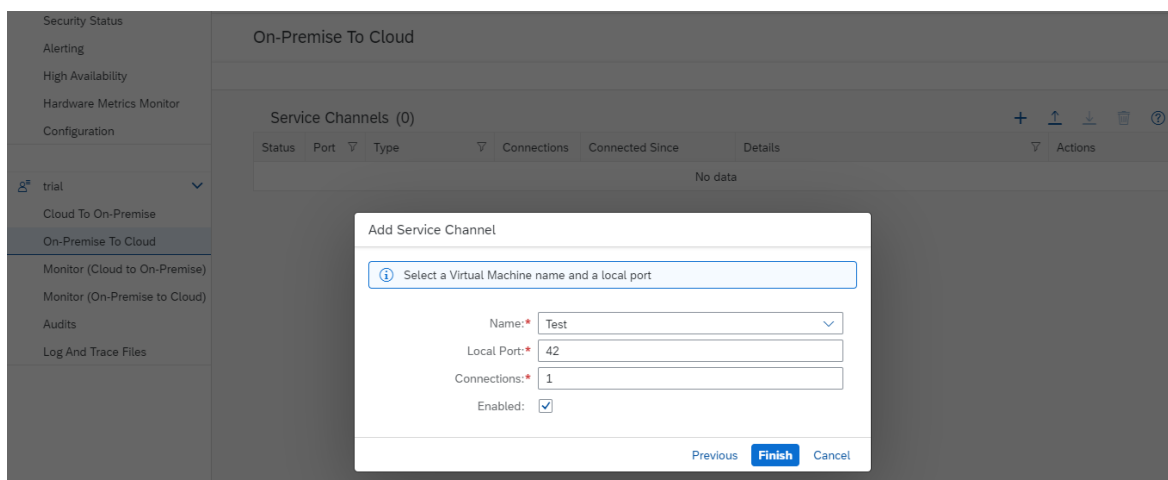


3. In the *Add Service Channel* dialog, select **Virtual Machine** from the list of supported channel types.
4. Optionally, provide a *Description* that explains what the Virtual Machine service channel is used for.
5. Choose *Next*. The *Virtual Machine* dialog opens.
6. Choose the Virtual Machine *<Name>* from the list of available Virtual Machines. It matches the corresponding name shown under *Virtual Machines* in the cockpit.

Note

The Virtual Machine name is case-sensitive.

7. Choose the *<Local Port>*. You can use any port that is not used yet.



8. Leave *<Enabled>* selected to establish the channel immediately after clicking *Save*. Unselect it if you don't want to establish the channel immediately.

9. Choose *Finish*.

Next Steps

Once you have established a service channel for the Virtual Machine, you can connect it with your SSH client. This may be done by accessing `<Cloud_connector_host>:<local_vm_port>` and the key file that was generated when creating the virtual machine.

1.5.2.7.4 Configure a Service Channel for RFC

For scenarios that need to call from on-premise systems to SAP BTP ABAP environment or to S/4HANA Cloud 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

S/4HANA Cloud

You have set up the S/4HANA Cloud environment for communication with the Cloud Connector.

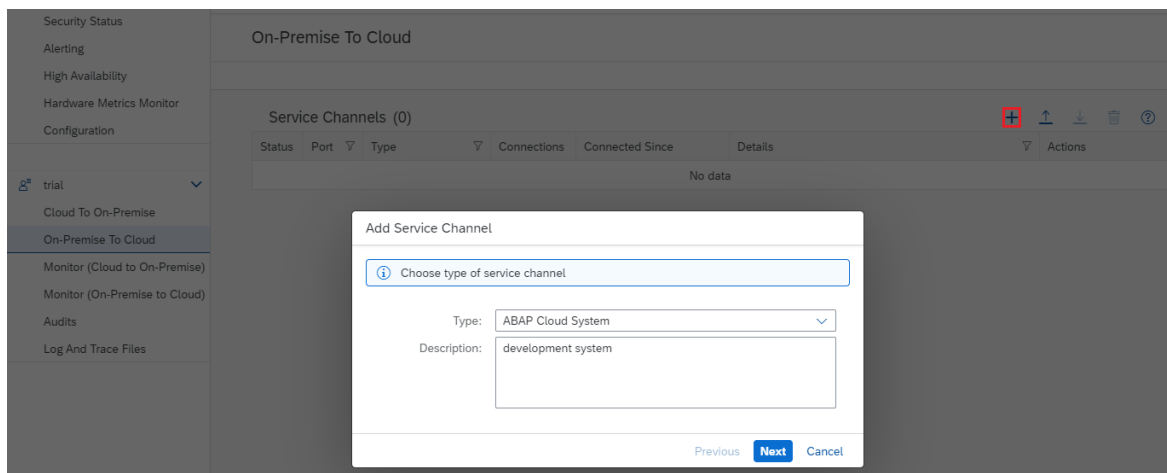
In particular, you must create a communication arrangement for the scenario SAP_COM_0200 (SAP Cloud Connector Integration). See [Integrating On-Premise Systems](#) (SAP S/4HANA Cloud documentation).

SAP BTP ABAP Environment

- 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, like for S/4HANA Cloud, a communication arrangement for the scenario SAP_COM_0200. For more information, see [Create a Communication Arrangement for Cloud Connector Integration](#) (documentation for ABAP environment on SAP BTP).

Procedure

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



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

Note

The S/4HANA Cloud tenant host name is case-sensitive. Also make sure that you specify the API address of your tenant host. For example, if the tenant host of your instance is **<my1234567>.s4hana.ondemand.com**, the API tenant host to be specified is **<my1234567>-api.s4hana.ondemand.com**.

7. In the same dialog window, define the **<Local Instance Number>** under which the ABAP Cloud system is reachable for the client systems. You can enter any instance number for which the port is not used yet on the Cloud Connector host. The port numbers result from the following pattern: **33<LocalInstanceNumber>**, for activated SNC (*Secure Network Connection*) **48<LocalInstanceNumber>**.
8. Use the checkbox *Activate SNC* to specify if the opened service channel port should listen to and terminate incoming SNC RFC connections instead of plain RFC connections. The SNC configuration used is the same as in section *Initial Configuration (RFC)* [page 287]. In case of issues, you can use the trace *Cloud Connector loggers*, and also the *SNC payload* and *CPIC* traces. For more information on traces, see *Troubleshooting* [page 580].

Note

This SNC option is only supported for ABAP-based clients, not for any RFC connectors as JCo, NCo or NW RFC SDK. For these scenarios, please use the WebSocket RFC protocol without the Cloud Connector.

9. In the same dialog window, leave *Enabled* selected to establish the channel immediately after choosing *Finish*. Unselect it if you don't want to establish the channel immediately.

10. Choose *Finish*.

Note

When addressing an ABAP Cloud system in a **destination configuration**, you must enter the *Cloud Connector host* as application server host. As instance number, specify the *<Local Instance Number>* that you configured for the service channel. As user, you must provide the *business user* name but not the technical user name associated with the same.

Note

When using an RFC service channel in a high availability setup, you can put a TCP load balancer between the client ABAP system and the Cloud Connector master and shadow instances. In this case, you must configure the load balancer in the destination. The load balancer routes the request to the instance that is currently in the master role, either by simply trying one instance after the other or by checking regularly, which of the instances currently has the master role, and routing to this instance only.

1.5.2.7.5 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 displays the SAP Cloud Connector Administration web interface. The top navigation bar includes the SAP logo, 'Cloud Connector Administration', and user information 'Administrator'. The left sidebar shows a navigation menu with options like 'Connector', 'Security Status', 'Alerting', 'High Availability', 'Hardware Metrics Monitor', 'Configuration', and a dropdown menu for 'trial'. The main content area is titled 'Connector' and includes buttons for '+ Add Subaccount', 'Backup', and 'Restore'. Below this, the 'Connector Overview' section shows details for a specific connector, including its ID, local name, local IP, security status (Low risk), high availability status (Disconnected), and the number of alerts (1). The 'Subaccount Dashboard (3)' section contains a table with columns for Status, Subaccount, Display Name, Location ID, Region, and Actions. The 'Service Channels Overview (3)' section contains a table with columns for Status, Port, Type, Subaccount, Details, and Actions.

Status	Subaccount	Display Name	Location ID	Region	Actions
	819986f5-460d-4a36-b...	qual	EMEA	Europe (Frankfurt) - AWS	
	ab1220dc-5bfb-45d1-9...	trial	EMEA	Trial	
	ztglf9vgi@	Virtual Machine Test	EMEA	Trial	

Status	Port	Type	Subaccount	Details	Actions
	1042	Virtual Machine	Virtual Machine Test	myVirtualMachine	
	3332	ABAP Cloud System	qual	my312345-api.s4hana.ondemand.com	
	31615	HANA Database	trial	abcd	

The [filter buttons](#) above the overview table let you filter the shown service channels based on their status. You can select all service channels, all enabled ones, all disabled ones, or all service channels for which enabling has failed.

1.5.2.8 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 \[page 501\]](#)

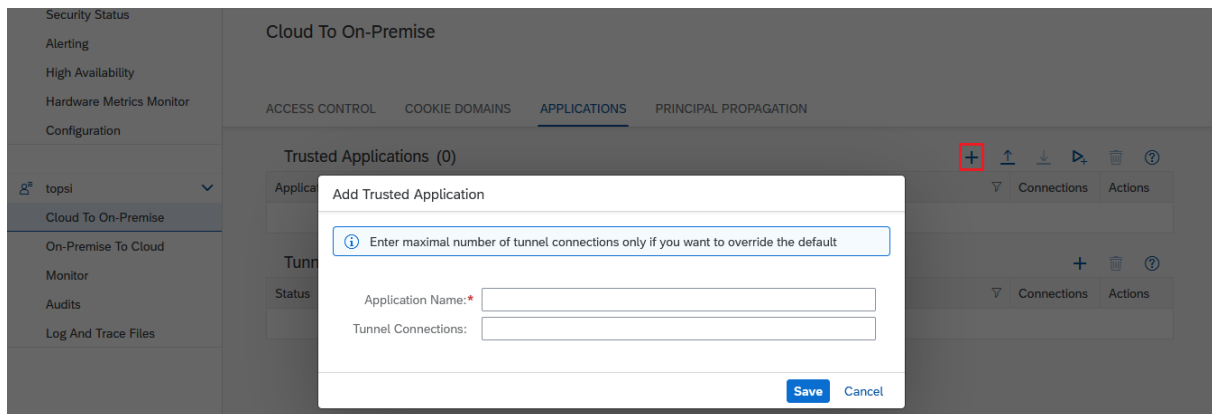
[Configure the Trust Store \[page 503\]](#)

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.

📘 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 \[page 507\]](#).

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

📘 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.

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**.

Security Status

Alerting

High Availability

Hardware Metrics Monitor

Configuration

d036325trial

Cloud To On-Premise

On-Premise To Cloud

Monitor

Audits

Log And Trace Files

Cloud To On-Premise

ACCESS CONTROL COOKIE DOMAINS APPLICATIONS PRINCIPAL PROPAGATION

Trusted Applications (0)

Application Name

White-list is empty — all applications will be trusted

Tunnel Connections Limits (1)

Status	Application Name	Connections	Actions
<input type="checkbox"/>	test	2	Edit Delete

Back to [Tasks \[page 501\]](#)

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:

Trust Store

No trusted backends are configured — HTTPS communication is not possible

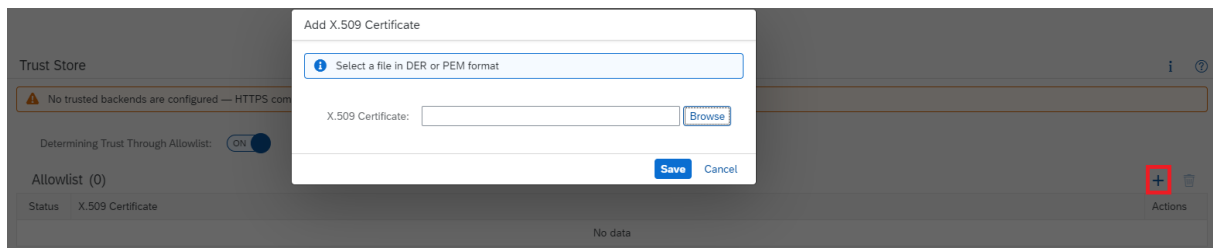
Determining Trust Through Allowlist: ☒ ON

Allowlist (0)

Status	Actions
X.509 Certificate	

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.

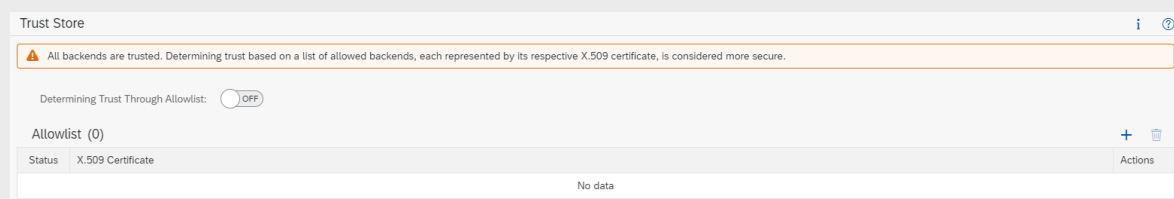


Note

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 \[page 501\]](#)

1.5.2.9 Configure Domain Mappings for Cookies

Context

Some HTTP servers return cookies that contain a `domain` attribute. For subsequent requests, HTTP clients should send these cookies to machines that have host names in the specified domain.

For example, if the client receives a cookie like the following:

```
Set-Cookie: cookie-field=some-value; domain=mycompany.corp; path=...; ...
```

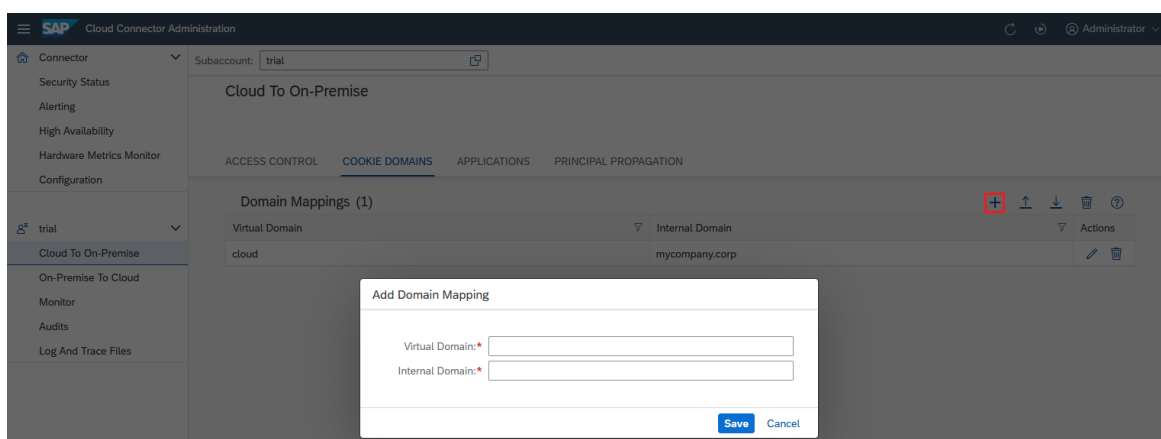
It returns the cookie in follow-up requests to all hosts like `ecc60.mycompany.corp`, `crm40.mycompany.corp`, and so on, if the other attributes like `path` and `attribute` require it.

However, in a Cloud Connector setup between a client and a Web server, this may lead to problems. For example, assume that you have defined a virtual host `sales-system.cloud` and mapped it to the internal

host name **ecc60.mycompany.corp**. The client "thinks" it is sending an HTTP request to the host name **sales-system.cloud**, while the Web server, unaware of the above host name mapping, sets a cookie for the domain **mycompany.corp**. The client does not know this domain name and thus, for the next request to that Web server, doesn't attach the cookie, which it should do. The procedure below prevents this problem.

Procedure

1. From your subaccount menu, choose **Cloud To On-Premise**, and go to the **Cookie Domains** tab.
2. Choose **Add**.
3. Enter **cloud** as the virtual domain, and your company name as the internal domain.
4. Choose **Save**.



The Cloud Connector checks the Web server's response for **Set-Cookie** headers. If it finds one with an attribute `domain=intranet.corp`, it replaces it with `domain=sales.cloud` before returning the HTTP response to the client. Then, the client recognizes the domain name, and for the next request against **www1.sales.cloud** it attaches the cookie, which then successfully arrives at the server on **machine1.intranet.corp**.

Note

Some Web servers use a syntax such as `domain=.intranet.corp` (RFC 2109), even though the newer RFC 6265 recommends using the notation without a dot.

Note

The value of the domain attribute may be a simple host name, in which case no extra domain mapping is necessary on the Cloud Connector. If the server sets a cookie with `domain=machine1.intranet.corp`, the Cloud Connector automatically reverses the **mapping** **machine1.intranet.corp** to **www1.sales.cloud** and replaces the cookie domain accordingly.

Related Information

[Configure Access Control \[page 340\]](#)

1.5.2.10 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.

Note

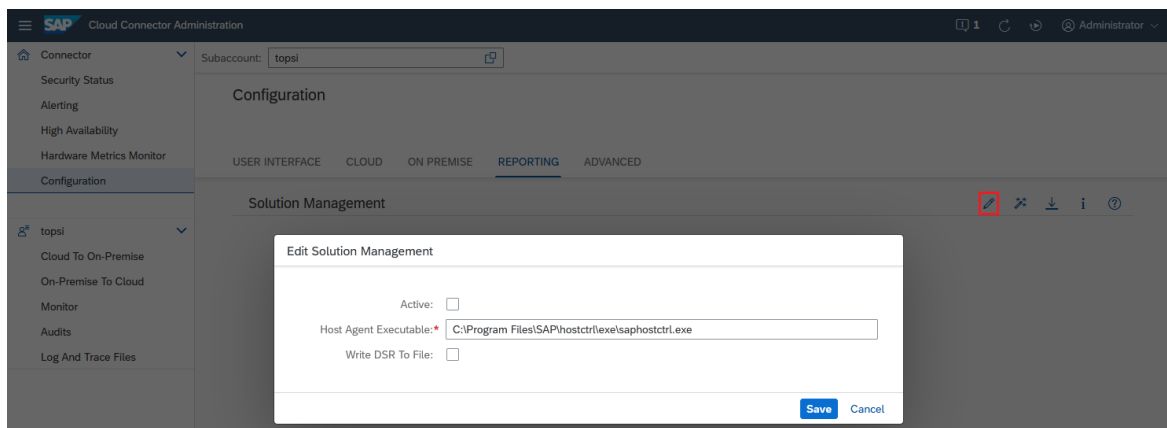
Linux OS: if you installed the host agent after installing the Cloud Connector, you can execute `enableSolMan.sh` in the installation directory (available as of Cloud Connector version 2.11.2) to adjust the host agent configuration and user group setup. This action requires root permission.

- The SAP Solution Manager must be of release 7.2 SP06 or higher.

Procedure

To enable the integration, do the following:

- From the Cloud Connector main menu, choose **Configuration** > **Reporting**. In section **Solution Management** of the **Reporting** tab, select **Edit**.



2. Select the [Active](#) checkbox.
3. In the field [Host Agent](#), specify the location of the host agent as filepath.
4. If you want to store the reporting results (*Dynamic Statistical Records*), select [Write DSR To File](#).
5. Choose [Save](#).

Note

To download the registration file *ImdbModel.xml*, choose the icon [Download registration file](#) from the [Reporting](#) tab.

Related Information

[Monitoring \[page 546\]](#)

1.5.2.11 Configure Advanced Connectivity

Adapt connectivity settings that control the throughput and HTTP connectivity to on-premise systems.

Tunnel Connections

If required, you can adjust the following parameters for the communication tunnel by changing their default values:

- Application Tunnel Connections (default: 1)

Note

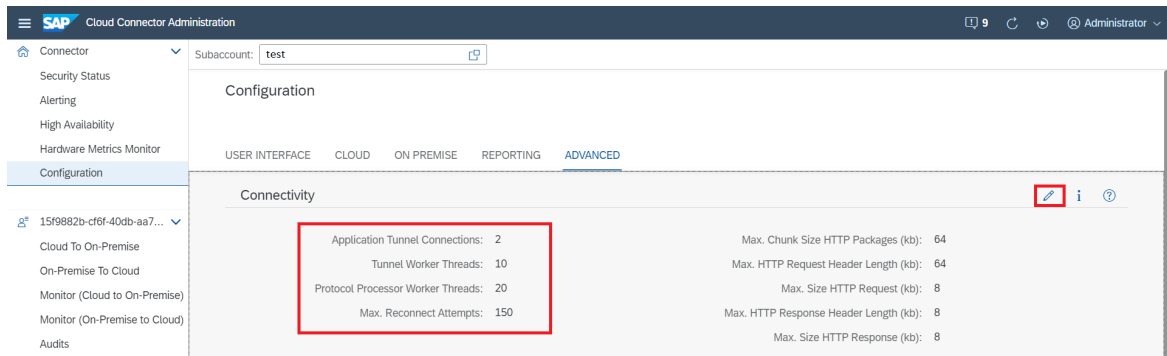
This parameter specifies the default value for the maximal number of tunnel connections *per application*. The value must be higher than 0.

- Tunnel Worker Threads (default: 10)
- Protocol Processor Worker Threads (default: 20)

For detailed information on connection configuration requirements, see [Configuration Setup \[page 262\]](#).

To change the parameter values, do the following:

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



2. In the **Edit Connectivity Settings** dialog, change the parameter values as required.
3. Choose **Save**.

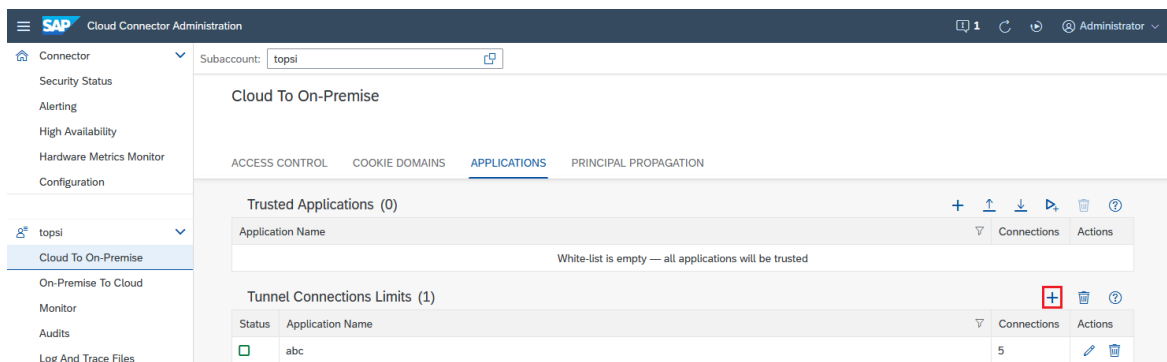
Additionally, you can specify the number of allowed tunnel connections for each application that you have specified as a [trusted application \[page 307\]](#).

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:

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



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

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

USER INTERFACE CLOUD ON PREMISE REPORTING **ADVANCED**

Connectivity

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

Find a brief description of these critical configuration parameters below:

Parameter	Description	Default Value
Max. Chunk Size HTTP Packages (kb)	Max. size of chunks transmitted in HTTP streaming. The chunk size affects the throughput of HTTP communication.	64kb
Max. HTTP Request Header Length (kb)	Max. allowed size of HTTP request headers. Headers containing authentication information like SAML or JWT could require this size.	64kb

Parameter	Description	Default Value
Max. Size HTTP Request (kb)	Size for the request line of HTTP request. HTTP Body is not included.	8kb
Max. HTTP Response Header Length (kb)	Max. allowed size of HTTP response headers.	8kb
Max. Size HTTP Response (kb)	Size for the response line of the HTTP response. HTTP Body is not included.	8kb

1.5.2.12 Configure the Java VM

Adapt the JVM settings that control memory management.

If required, you can adjust the following parameters for the Java VM by changing their default values:

- Initial Heap Size (default: 1024 MB)
- Maximal Heap Size (default: 1024 MB)
- Maximal Direct Memory (default: 2 GB)

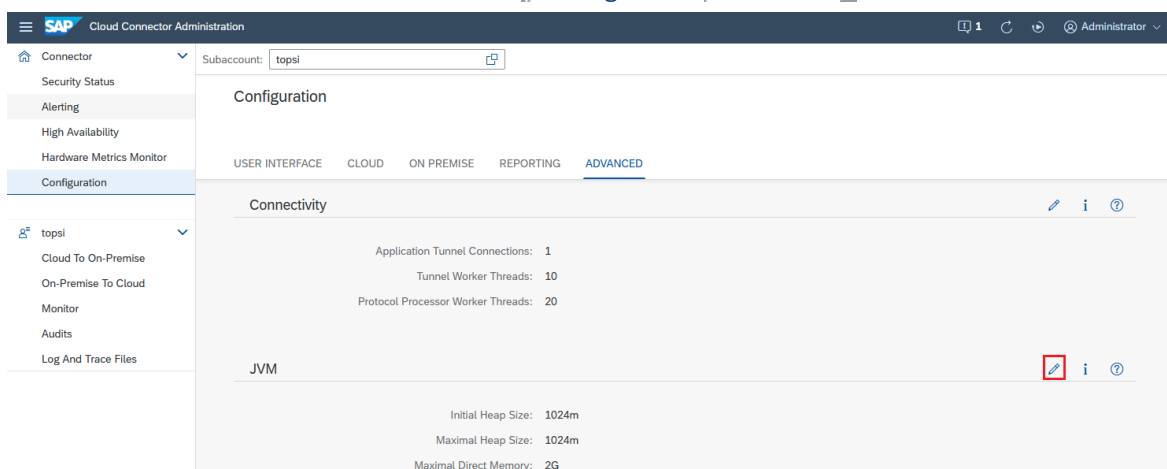
Note

A restart is required when changing JVM settings.

We recommend that you set the initial heap size equal to the maximal heap size, to avoid memory fragmentation.

To change the parameter values, do the following:

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



2. In the **Edit JVM Settings** dialog, change the parameter values as required.
3. Choose **Save**.

Related Information

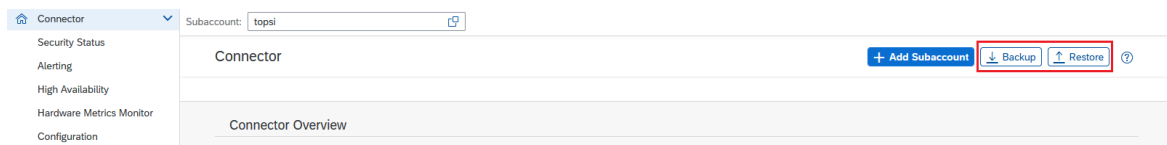
[Sizing for the Master Instance \[page 259\]](#)

1.5.2.13 Configuration Backup

You can backup and restore your Cloud Connector configuration.

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

1. From the Cloud Connector main menu, choose [Connector](#).



2. To backup or restore your configuration, choose the respective icon in the upper right corner of the screen.
 1. To backup your configuration, enter and repeat a password in the [Backup](#) dialog and choose [Backup](#).

Note

An archive containing a snapshot of the current Cloud Connector configuration is created and downloaded by your browser. For security reasons, some files are encrypted, using the password provided for the backup procedure.

Tip

You can use this archive to restore the current state of the same installation or to move the current configuration to a new Cloud Connector installation, if the original instance can no longer be used.

You can restore the archive on a Cloud Connector instance

- With the same or higher version than the original one.
- Running on a different operating system.

However, you cannot restore it on an *older* Cloud Connector version.

2. To restore your configuration, enter the required *Archive Password* and the *Login Password* of the currently logged-in administrator user in the [Restore from Archive](#) dialog and choose [Restore](#).

Caution

The restore action overwrites the current configuration of the Cloud Connector. Its current state and settings will be lost permanently unless you have created another backup before doing the restore operation. Upon successfully restoring the configuration backup, the Cloud Connector restarts automatically. All active sessions are then terminated.

Note

- The `props.ini` file is treated in a special way. If the file from the backup archive differs from the one that is used in the current installation, it will be placed next to the original one as

`props.ini.restored`. If you want to use the `props.ini.restored` file, replace the existing `props.ini` file by the restored one on OS level and restart the Cloud Connector afterwards.

- All custom path settings from the configuration backup archive, which are not valid or operable anymore, will be automatically reset to their respective default values upon restore operation. For example, this will be the case if a configured directory or file path from the backup archive does not exist or is not accessible anymore.
- Restored certificates from older backup archives might have expired and must be replaced with valid ones to get all configured scenarios to work again.
- If restoring a configuration backup archive on a host that is different from the original one, the restored UI certificate might not be issued for the new hostname, domain or IP address, and therefore would require a replacement with a matching UI server certificate as well.
- If you are using an external SNC library, neither the library itself is part of the configuration backup archive, nor any configuration that this library would load from its own external storage. As a result, the SNC configuration must be set up again after restoring a configuration backup archive on a different host.
- The same applies to used own CA certificates which are stored in the JVM's managed trust store, for example, if Email alerting or LDAP has been set up this way, with own certificates for establishing a secure communication. These externally stored CA certificates would also not be part of the configuration backup archive and must be imported into the local JVM's trust store manually.







Caution

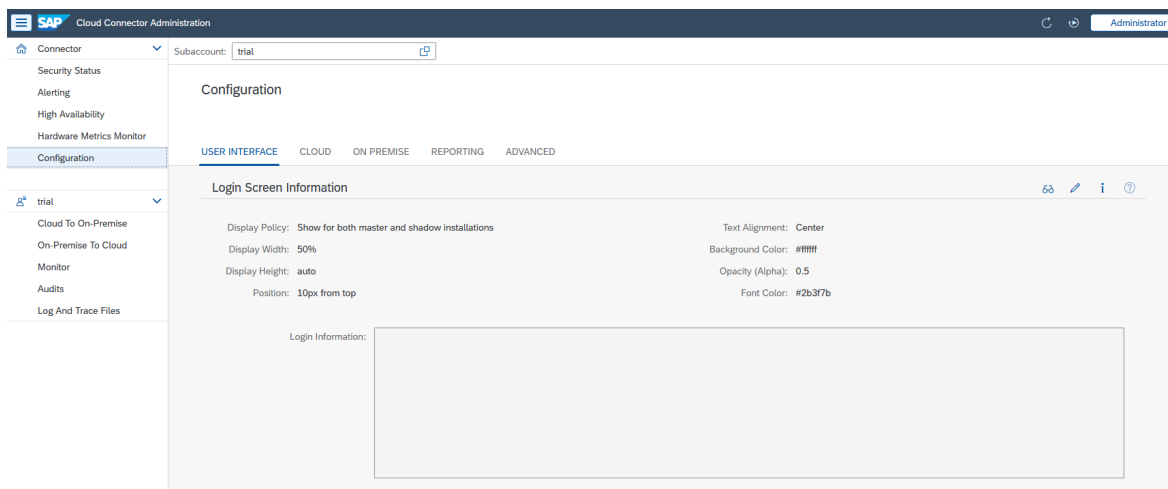
Do not run multiple Cloud Connector instances with the same configuration simultaneously. The feature to restore a configuration backup archive on another Cloud Connector installation is not supposed to be used for cloning purposes, but only for supporting the move of a Cloud Connector instance from one host to another. After restoring a configuration backup archive on a different Cloud Connector installation, you must stop the original Cloud Connector instance *immediately*. Otherwise, running two or more instances with the same configuration will cause issues. Such a cloned setup is not supported.

1.5.2.14 Configure Login Screen Information

Add additional information to the login screen and configure its appearance.

To configure the login screen information, proceed as follows:

1. Go to  [Configuration](#)  [User Interface](#)  (Shadow instance:  [Shadow Configuration](#)  [User Interface](#) ) and press the [Edit](#) button in section [Login Screen Information](#) (at the bottom of the screen).



As a result, the following dialog is opened:

Edit Login Information

Display Policy

- ☒ Show for both master and shadow installations ☐ Show only if this is a master installation
☐ Show only if this is a shadow installation ☐ Do not show at all

Login Display Properties



Display Width:	<input type="text" value="50%"/>	Background Color:	<input type="text" value="#ffffff"/>
Display Height:	<input type="text" value="auto"/>	Opacity (Alpha):	<input type="text" value="0.5"/>
Text Alignment:	<input type="text" value="Center"/>	Font Color:	<input type="text" value="#2b3f7b"/>
Position:	<input type="text" value="10px"/>	<input checked="" type="radio"/> from top	<input type="radio"/> from bottom

Login Information



Enter an HTML fragment to be displayed as login information. Restrictions apply. Compliance with these restrictions will be checked when saving. Consult the documentation for details.

Save Cancel

- In section [Display Policy](#), select a display policy for the login screen information. The display policy decides whether the information is shown if the instance is assuming the role mentioned in the policy.
- In section [Display Properties](#), specify your preferred display properties (appearance and position):
 - The login information is displayed in a box with rounded corners. You can specify its width and height in pixels (unit px) or as a percentage (unit %).

Note

Omitting any value triggers the default or auto behavior.

- For the width, the default behavior is equivalent to 100%.
- For the height, the default behavior sets the height to a value that accommodates the login information (that is, the given HTML fragment). For extensive information we therefore

recommend that you limit the height to a suitable pixel or percentage value to induce scrolling, and to prevent the box from growing beyond a reasonable height.

- When customizing the **background color** of the box, the **opacity** (of the background color), **font color**, or **text alignment**, then the section [Login Information](#) automatically switches to preview mode. That way you can follow live the changes made to the appearance of the box and of the information displayed inside it.

Note

You can hide the box and to show only the text of the login information by choosing an opacity value of **0** (opacity is the opposite of transparency. *No* opacity means *complete* transparency).

- You can **position the box** containing the login information at the top or bottom of the login page. To do this, set the field `<Position>` to the corresponding pixel or percentage value.
4. Enter the information to be displayed in section [Login Information](#). The information must be supplied as an HTML fragment. There is a limited number of tags that can be used. Attributes available for these tags are subject to restrictions.

Available Tag	Attribute Restriction
p	Only attribute style is permitted, with property text-align set to a valid value (left , right , center , or justified)
ul	No attributes allowed
ol	No attributes allowed
li	No attributes allowed
br	No attributes allowed
h1	No attributes allowed
h2	No attributes allowed
h3	No attributes allowed
i	No attributes allowed
b	No attributes allowed
a	Must have exactly two attributes: <ul style="list-style-type: none">href (its value must be a <code><URL></code> with protocol http or https)target (its value must be "_blank")

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

Note

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

1.5.3 Administration

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

Topic	Description
Exchange UI Certificates in the Administration UI [page 517]	By default, the Cloud Connector includes a self-signed UI certificate. It is used to encrypt the communication between the browser-based user interface and the Cloud Connector itself. For security reasons, however, you should replace this certificate with your own one to let the browser accept the certificate without security warnings.
Logon to the Cloud Connector via Client Certificate	Switch from default logon to client certificate logon to access the Cloud Connector.
Configure Named Cloud Connector Users [page 519]	If you operate an LDAP server in your system landscape, you can configure the Cloud Connector to use the named users who are available on the LDAP server instead of the default Cloud Connector users.
High Availability Setup [page 537]	The Cloud Connector lets you install a redundant (shadow) instance, which monitors the main (master) instance.
Change the UI Port [page 543]	Use the changeport tool (Cloud Connector version 2.6.0+) to change the port for the Cloud Connector administration UI.
Connect and Disconnect a Cloud Subaccount [page 544]	As a Cloud Connector administrator, you can connect the Cloud Connector to (and disconnect it from) the configured cloud subaccount.
Secure the Activation of Traffic Traces [page 545]	Tracing of network traffic data may contain business critical information or security sensitive data. You can implement a "four-eyes" (double check) principle to protect your traces (Cloud Connector version 1.3.2+).
Monitoring [page 546]	Use various views to monitor the activities and state of the Cloud Connector.
Alerting [page 573]	Configure the Cloud Connector to send email alerts whenever critical situations occur that may prevent it from operating.
Audit Logging [page 576]	Use the auditor tool to view and manage audit log information (Cloud Connector version 2.2+).
Troubleshooting [page 580]	Information about monitoring the state of open tunnel connections in the Cloud Connector. Display different types of logs and traces that can help you troubleshoot connection problems.

Topic	Description
Process Guidelines for Hybrid Scenarios [page 585]	How to manage a hybrid scenario, in which applications running on SAP BTP require access to on-premise systems using the Cloud Connector.
Configuring Backup [page 588]	Find an overview of backup procedures for the Cloud Connector.

1.5.3.1 Exchange UI Certificates in the Administration UI

By default, the Cloud Connector includes a self-signed UI certificate. It is used to encrypt the communication between the browser-based user interface and the Cloud Connector itself. For security reasons, however, you should replace this certificate with your own one to let the browser accept the certificate without security warnings.

Procedure

Master Instance

1. From the main menu, choose [Configuration](#) and go to the [User Interface](#) tab.
2. In the [UI Certificate](#) section, start a certificate signing request procedure by choosing the icon [Generate a Certificate Signing Request](#).
3. In the pop-up [Generate CSR](#), specify a key size and a subject fitting to your host name.
For host matching, you should use the available names within the `subjectAlternativeName` (SAN) extension, see [RFC 2818](#) and <https://www.chromestatus.com/feature/4981025180483584>. A check verifies whether the host matches one of the entries in the SAN extension.
In section **Subject Alternative Names**, you can add additional values by pressing the [Add](#) button. Choose one or more of the following SAN types and provide the matching values:
 - DNS: a specific host name (for example, `www.sap.com`) or a wildcard hostname (for example, `*.sap.com`).
 - IP: an IPv4 or IPv6 address.
 - [RFC822](#): an example for this type of value is a simple email address: for example, `donotreply@sap.com`.
 - URI: a URI for which the certificate should be valid.

Generate CSR

Key Size

☐ 2048 Bits
☒ 4096 Bits

Subject DN

Common Name (CN): *

scc.internal.corp

E-Mail Address (EMAIL):

Locality (L):

Organizational Unit (OU):

department

Organization (O):

companyName

State or Province (ST):

Country (C):

DE

Subject Alternative Names

+

Type	Value	Actions
RFC822	ownerscc@company.com	
DNS	*.internal.corp	

Generate

Cancel

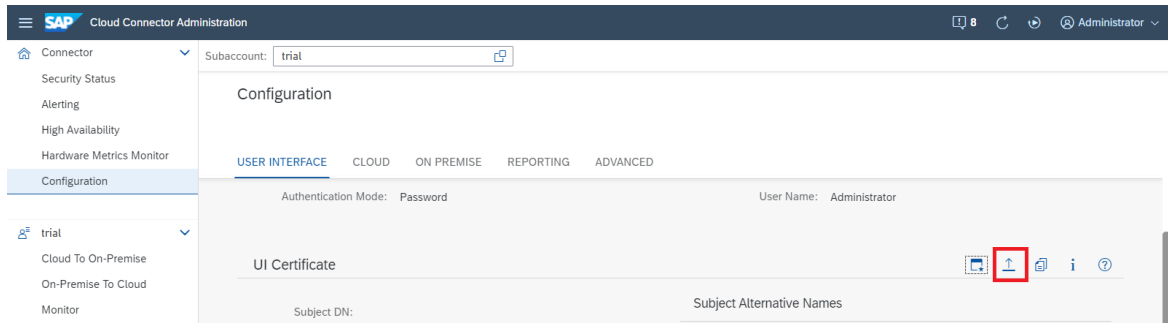
- Press [Generate](#).
- You are prompted to save the signing request in a file. The content of the file is the signing request in PEM format.

The signing request must be provided to a Certificate Authority (CA) - either one within your company or another one you trust. The CA signs the request and the returned response should be stored in a file.

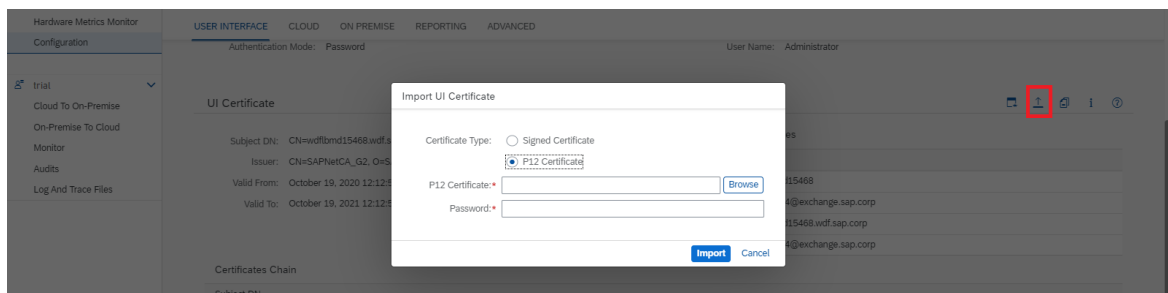
Note

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

- To import the signing response, choose the [Upload](#) icon.



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



7. Select [Browse](#) to locate the file and then choose the [Import](#) button.
8. Review the certificate details that are displayed.
9. Restart the Cloud Connector to activate the new certificate.

Shadow Instance

In a High Availability setup, perform the same operation on the shadow instance.

⚠ Caution

UI certificates are used for the secure communication between master and shadow instances. Replacing the UI certificate breaks the trust relationship and communication between master and shadow is not possible anymore.

Please disconnect the shadow instance when you are going to replace UI certificate(s). Once the certificate update is done, connect the shadow instance again. You will be forced to enter user and password again to establish the trust relationship between master and shadow instances.

1.5.3.2 Configure Named Cloud Connector Users

Set up LDAP-based user management for the Cloud Connector.

LDAP-Based User Management

We recommend that you configure LDAP-based user management for the Cloud Connector to allow only named administrator users to log on to the administration UI.

This guarantees traceability of the Cloud Connector configuration changes via the Cloud Connector audit log. If you use the default and built-in `Administrator` user, you cannot identify the actual person or persons who perform configuration changes. Also, you will not be able to use different types of user groups.

Configuration

If you have an LDAP server in your landscape, you can configure the Cloud Connector to authenticate Cloud Connector users against the LDAP server.

Valid users or user groups must be assigned to one of the following roles:

- Administrator users: `admin` or `sccadmin`
- Display users: `sccdisplay` or `sccmonitoring`

Note

The role `sccmonitoring` provides access to the monitoring APIs, and is particularly used by the SAP Solution Manager infrastructure, see [Monitoring APIs \[page 556\]](#).

- Support users: `sccsupport`

Alternatively, you can define custom role names for each of these user groups, see: [Use LDAP for User Administration \[page 521\]](#).

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

Related Information

[Use LDAP for User Administration \[page 521\]](#)

[Logon to the Cloud Connector via Client Certificate](#)

1.5.3.2.1 Use LDAP for User Administration

You can use LDAP (Lightweight Directory Access Protocol) to manage Cloud Connector users and authentication.

After installation, the Cloud Connector uses file-based user management by default. Alternatively, the Cloud Connector also supports LDAP-based user management. If you operate an LDAP server in your landscape, you can configure the Cloud Connector to use the LDAP user base.

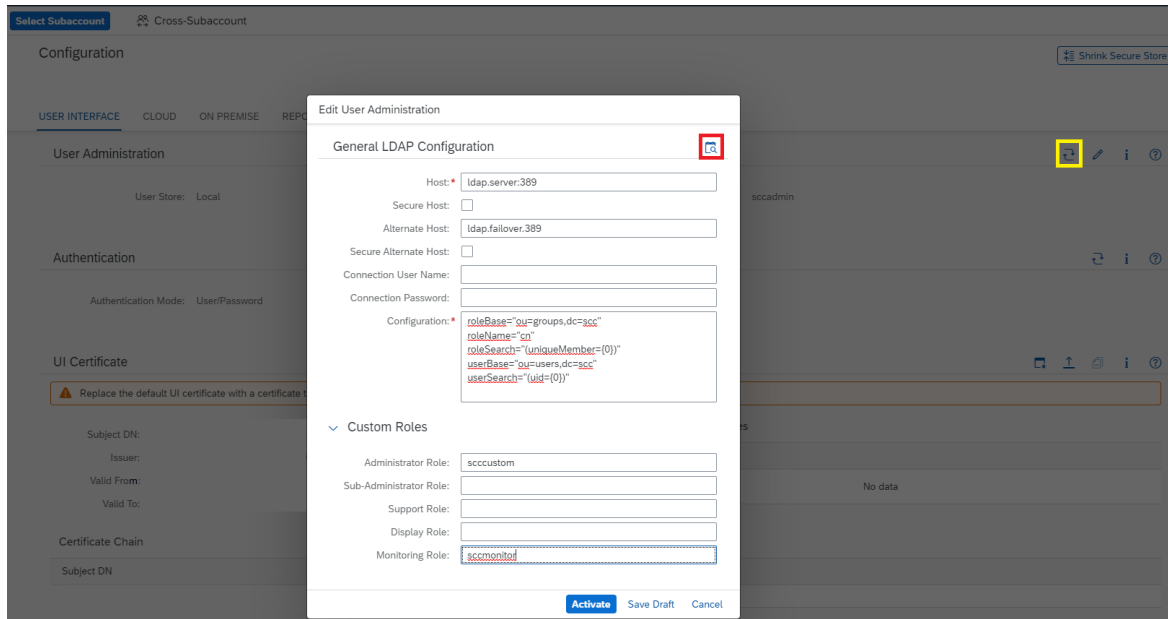
If LDAP authentication is active, you can assign users or user groups to the following default roles:

User Role	Authorization
sccadmin or admin	Administrate the Cloud Connector (all CRUD operations).
sccsubadmin	<ul style="list-style-type: none">• Manage all subaccount-related settings.• Perform support-related tasks like setting trace levels or creating a thread dump. Access to common settings for all subaccounts, like system certificate settings, is permitted.
sccdisplay	Access the Cloud Connector administration UI in read-only mode.
sccsupport	<ul style="list-style-type: none">• Access the Cloud Connector administration UI in read-only mode.• Perform support-related tasks like setting trace levels or creating a thread dump.
sccmonitoring	Provides access to the monitoring APIs, and is particularly used by the SAP Solution Manager infrastructure, see Monitoring APIs [page 556] .

Group membership is checked by the Cloud Connector.

Setting LDAP Authentication

1. From the main menu, choose [Configuration](#) and go to the [User Interface](#) tab.
2. From the [User Administration](#) section, choose [Switch to LDAP](#).



3. (Optional) To save intermediate adoptions of the LDAP configuration, choose *Save Draft*. This lets you store the changes in the Cloud Connector without activation.
4. Usually, the LDAP server lists users in an LDAP node and user groups in another node. In this case, you can use the following template for LDAP configuration. Copy the template into the configuration text area:

```
roleBase="ou=groups,dc=scc"
roleName="cn"
roleSearch="(uniqueMember={0})"
userBase="ou=users,dc=scc"
userSearch="(uid={0})"
```

Change the `<ou>` and `<dc>` fields in `userBase` and `roleBase`, according to the configuration on your LDAP server, or use some other LDAP query.

Note

The configuration depends on your specific LDAP server. For details, contact your LDAP administrator.

5. Provide the LDAP server's host and port (port **389** is used by default) in the `<Host>` field. To use the secure protocol variant LDAPS based on TLS, select *Secure*.
6. Provide a failover LDAP server's host and port (port **389** is used by default) in the `<Alternate Host>` field. To use the secure protocol variant LDAPS based on TLS, select *Secure Alternate Host*.
7. (Optional) Depending on your LDAP server configuration you may need to specify the `<Connection User Name>` and its `<Connection Password>`. LDAP Servers supporting anonymous binding ignore these parameters.
8. (Optional) To use your own role names, you can customize the default role names in the *Custom Roles* section. If no custom role is provided, the Cloud Connector checks permissions for the corresponding default role name:
 - `<Administrator Role>` (default: sccadmin)
 - `<Support Role>` (default: sccsupport)
 - `<Display Role>` (default: sccdisplay)
 - `<Monitoring Role>` (default: sccmonitoring)

9. (Optional) Before activating the LDAP authentication, you can execute an authentication test by choosing the [Test LDAP Configuration](#) button. In the pop-up dialog, you must specify user name and password of a user who is allowed to logon after activating the configuration. The check verifies if authentication would be successful or not.

Note

We strongly recommend that you perform an authentication test. If authentication should fail, login is not possible anymore. The test dialog also provides a test protocol, which could be helpful for troubleshooting.

For more information about how to set up LDAP authentication, see tomcat.apache.org/tomcat-8.5-doc/realms-howto.html .

Note

To find a list of all supported attributes, see https://tomcat.apache.org/tomcat-8.5-doc/config/realms.html#JNDI_Directory_Realm_-_org.apache.catalina.realm.JNDIRealm .

You can also configure LDAP authentication on the shadow instance in a high availability setup (master and shadow). From the main menu of the shadow instance, select [Shadow Configuration](#), go to tab [User Interface](#), and check the [User Administration](#) section.

Note

If you are using LDAP together with a high availability setup, you cannot use the configuration option `userPattern`. Instead, use a combination of `userSearch`, `userSubtree` and `userBase`.

Caution

An LDAP connection over TLS can cause TLS errors if the LDAP server uses a certificate that is not signed by a trusted CA. If you cannot use a certificate signed by a trusted CA, you must set up the trust relationship manually, that is, import the public part of the issuer certificate to the JDK's trust storage.

Usually, the `cacerts` file inside the java directory (`jre/lib/security/cacerts`) is used for trust storage. To import the certificate, you can use `keytool`:

```
keytool -import -storepass changeit -file <certificate used by LDAP server> -keystore cacerts -alias <e.g. LDAP_xyz>
```

For more information, see also <https://docs.oracle.com/cd/E19830-01/819-4712/ablqw/index.html> .

10. After finishing the configuration, choose [Activate](#). Immediately after activating the LDAP configuration you must restart the Cloud Connector server, which invalidates the current browser session. Refresh the browser and logon to the Cloud Connector again, using the credentials configured at the LDAP server.
11. To switch back to file-based user management, choose the [Switch](#) icon in section [User Administration](#) again.

Note

If you have set up an LDAP configuration incorrectly, you may not be able to logon to the Cloud Connector again. In this case, adjust the Cloud Connector configuration to use the file-based user store again *without the administration UI*. For more information, see the next section.

Switching Back to File-Based User Store without the Administration UI

If your LDAP settings do not work as expected, you can use the `useFileUserStore` tool, provided with Cloud Connector version 2.8.0 and higher, to revert back to the file-based user store:

1. Change to the installation directory of the Cloud Connector and enter the following command:
 - **Microsoft Windows:** `useFileUserStore`
 - **Linux, Mac OS:** `./useFileUserStore.sh`
2. Restart the Cloud Connector to activate the file-based user store.

For versions older than 2.8.0, you must manually edit the configuration files.

Depending on your operating system, the configuration file is located at:

- **Microsoft Windows OS:**
`<install_dir>\config_master\org.eclipse.gemini.web.tomcat\default-server.xml`
- **Linux OS:** `/opt/sap/scc/config_master/org.eclipse.gemini.web.tomcat/default-server.xml`
- **Mac OS X:** `/opt/sap/scc/config_master/org.eclipse.gemini.web.tomcat/default-server.xml`

1. Replace the Realm section with the following:

```
<Realm className="org.apache.catalina.realm.LockOutRealm">
  <Realm className="org.apache.catalina.realm.CombinedRealm">
    <Realm
      X509UsernameRetrieverClassName="com.sap.scc.tomcat.utils.SccX509SubjectDnRetriever"
      className="org.apache.catalina.realm.UserDatabaseRealm"
      digest="SHA-256" resourceName="UserDatabase" />
    <Realm
      X509UsernameRetrieverClassName="com.sap.scc.tomcat.utils.SccX509SubjectDnRetriever"
      className="org.apache.catalina.realm.UserDatabaseRealm" digest="SHA-1"
      resourceName="UserDatabase" />
    </Realm>
  </Realm>
```

2. Restart the Cloud Connector service:
 - **Microsoft Windows OS:** Open the Windows [Services](#) console and restart the cloud connector service.
 - **Linux OS:** Execute
 - System V init distributions: `service scc_daemon restart`
 - Systemd distributions: `systemctl restart scc_daemon`
 - **Mac OS X:** Not applicable because no daemon exists (for Mac OS X, only a portable variant is available).

Related Information

[LDAP Configuration: Best Practices \[page 525\]](#)

1.5.3.2.1.1 LDAP Configuration: Best Practices

Get background information on LDAP configuration for the Cloud Connector.

[Introduction \[page 525\]](#)

[Connect to the LDAP Server \[page 525\]](#)

[TLS Issues \[page 527\]](#)

[Authentication \[page 527\]](#)

[User Selection \[page 530\]](#)

[User Roles \[page 533\]](#)

[Relationship between User and Group \[page 534\]](#)


[Custom User Roles \[page 536\]](#)

[Additional Notes \[page 537\]](#)

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 . It explains the LDAP configuration parameters and considers various LDAP directory setups, including their specific configuration.

However, some aspects may raise questions. For this reason, we show you how to configure LDAP and verify LDAP configuration, providing useful background information in this topic.

A basic understanding of LDAP and tomcat's how-to guide is a prerequisite. As help tool, we are using the *ldapsearch* utility. You can use any LDAP client for this procedure.

Back to [Top \[page 525\]](#)

Connect to the LDAP Server

In a first step, you must establish a connection to the LDAP server. Like an HTTP connection, the connection to LDAP can be secure (via TLS) or plain. It points to a host and port. The address looks like this:

- TLS connection: `ldaps://<ldap.server.in.your.company>:<numeric port>`

- Plain connection: `ldap://<ldap.server.in.your.company>:<numeric port>`

Sample Code

```
ldapsearch -H ldap://<ldaphost>:<port>
```


The return value is `-1` if the address is not reachable. Before you go ahead, you need to know the address of your LDAP server. As soon as the `ldapsearch` utility returns a value other than `-1`, the address of the LDAP server is correct. More precisely, it indicates only that there is a server listening on this port, which is supposed to be the LDAP server.

Once the address is known, you can test the connection in the Cloud Connector. Enter the address and add a dummy configuration, for example, `x="x"`, to outwit the check. Then choose the test icon in the upper right corner. For a valid address, the LDAP configuration test in the Cloud Connector reports the following:

ADVANCED

Edit Authentication

✖ LDAP server authentication failed →Details

General LDAP Configuration 

Host: *

Secure Host: ☐

Alternate Host:

Secure Alternate Host: ☐

Connection User Name:

Connection Password:

Configuration: * `x="x"`

> Custom Roles

Test Results

Host:
Reachable

Authentication:
Failed

Details:

- Connecting to URL `ldap://wdfbmd15468.wdf.sap.corp:10389`
- Exception performing authentication. Retrying... throwing exception `javax.naming.AuthenticationException: [LDAP: error code 49 - INVALID_CREDENTIALS: Bind failed: ERR_229 Cannot authenticate user]`
- Connecting to URL `ldap://wdfbmd15468.wdf.sap.corp:10389`
- Exception performing authentication throwing exception `javax.naming.AuthenticationException: [LDAP: error code 49 - INVALID_CREDENTIALS: Bind failed: ERR_229 Cannot authenticate user]`
- Returning null principal.
- Authentication for user c failed

What You Can Do:
Verify connection user and password

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

Administrator

Activate Save Draft Cancel

The message: `Connecting to URL ldap://<ldaphost>:<port>; Exception performing authentication` indicates that your LDAP server requires user and password for queries.

Back to [Top \[page 525\]](#)

TLS Issues

LDAP connection over TLS will run into TLS errors if the LDAP server uses an "untrusted" certificate. This could be a self-signed certificate or a certificate signed by a generally untrusted authority.

If you cannot use a trusted certificate on your LDAP server, you must import the public part of the issuer certificate to the JDK's trust storage. See the JDK documentation how to do that.

Usually, the trust storage location is `cacerts` inside the java directory (`jre/lib/security/cacerts`). You can use the `keytool` utility for import.

Sample Code

```
keytool -import -storepass changeit -file <certificate used by LDAP server>
-keystore cacerts -alias <e.g. LDAP_xyz>
```

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

Back to [Top \[page 525\]](#)

Authentication

If the address of the LDAP server is correct and the Cloud Connector can establish a connection, you can proceed with the next step: the authentication by the LDAP directory.

The LDAP server may require authentication or not (anonymous connection), before a query can be executed. Authentication is done by user and password, specified by the `connectionName` and `connectionPassword` properties.

Anonymous access is sufficient in most cases and provides the same level of security. However, you have to deal with the existing setup on the LDAP server.

Note

The LDAP user is not the same user that is later used to logon on to the Cloud Connector. It is a specific user, which has permissions to query the LDAP directory. It can be stored in an LDAP directory separated from other users. We recommend that you specify the fully-qualified user name like `"uid=admin,ou=system"`.

Additionally, *MS Active Directory* allows authentication for `user@domain`, for example, `ldapUser@company.local`.

To verify the values, let's first check the authentication with `ldapsearch`:

🔗 Sample Code

```
ldapsearch -D "uid=admin,ou=system" -w <password> -H ldap://<ldaphost>:<port>  
-b "x=x"
```

📌 Note

We added a non-existing user base `-b "x=x"` to prevent long output.

The only thing to check here is, if authentication is ok. If it is not, LDAP returns *INVALID_CREDENTIALS: Bind failed*:

🔗 Sample Code

```
ldapsearch -D "uid=admin,ou=system" -w <wrong password> -H ldap://  
<ldaphost>:<port> -b "x=x"  
ldap_bind: Invalid credentials (49)  
additional info: INVALID_CREDENTIALS: Bind failed: ERR_229 Cannot  
authenticate user uid=admin,ou=system
```

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

Edit Authentication

General LDAP Configuration 🔍

***Host:**

Secure Host: ☐

Alternate Host:

Secure Alternate Host: ☐

Connection User Name:

Connection Password:

***Configuration:**

▼ **Custom Roles**

Administrator Role:

Support Role:

Display Role:

Monitoring Role:

[Save Draft](#) [Activate](#) [Cancel](#)

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

Edit Authentication

❗ Authentication failed [→Details](#)

General LDAP Configuration 🔍

***Host:**

Secure Host: ☐

Alternate Host:

Secure Alternate Host: ☐

Connection User Name:

Connection Password:

Test Results

Host:
Reachable

Authentication:
Failed

Details:

- Connecting to URL ldap://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 \[page 525\]](#)

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 login.

`userSearch` selects the attribute containing the user ID. Together with the `userBase`, the configuration looks like this now:

Sample Code

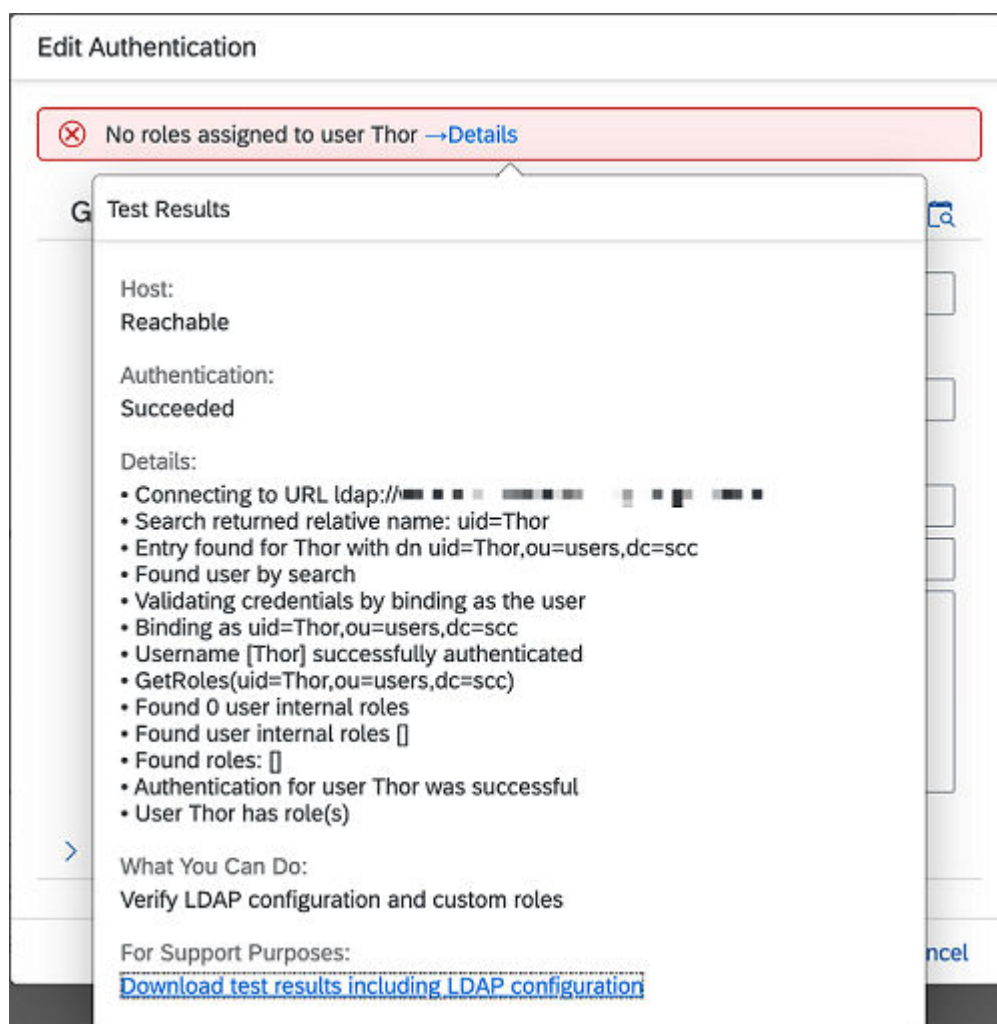
```
userBase="ou=users,dc=scc"
userSearch="(uid={0})"
```

The corresponding LDAP selection is:

Sample Code

```
ldapsearch -D "uid=admin,ou=system" -w <password> -H ldap://<ldaphost>:<port>  
-b "ou=users,dc=scc" "(&(uid=Thor))"
```

Now, we take a closer look now at the response. The user *Thor* was found, its password was successfully validated by LDAP server, but there are no specific roles selected:



Let's assume that the users are not located under the same branch in the LDAP tree. For this case, you cannot define more than one *userBase*. Instead, you can set *userBase* for the corresponding parent node, which then includes all the user branches. To achieve this, add *userSubtree="true"* to the configuration.

ANCED

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

Edit Authentication

No roles assigned to user Thor →Details

General LDAP Configuration

Host:*

Secure Host: ☐

Alternate Host:

Secure Alternate Host: ☐

Connection User Name: uid=admin,ou=system

Connection Password: ****

Configuration:*

userSubtree="true"
userBase="dc=scc"
userSearch="(uid={})"

> Custom Roles

Activate

Save Draft

Cancel

Taking a look at the relative name returned by search, it is `uid=Thor,ou=users` now.

Besides `uid`, you are free to use every other attribute of the user node. For example, for *Active Directory*, the preferred attribute is often `sAMAccountName`, the corresponding configuration is `userSearch="(sAMAccountName={0})"`.

ⓘ Note

For *Active Directory*, it might be necessary to add `adCompat="true"` to the configuration.

As mentioned above, you can use every attribute as user ID as long as it is *unique*. To verify this, check the query result for the respective attribute with *Idapsearch*. 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=scs" "(&(cn=<userid>))"
```


Note

If `connectionUser` is located under `userBase` and its ID can be selected by the same `userSearch`, you can use just the user ID in the `connectionUser` field instead of a fully-qualified DN.

Back to [Top \[page 525\]](#)

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.

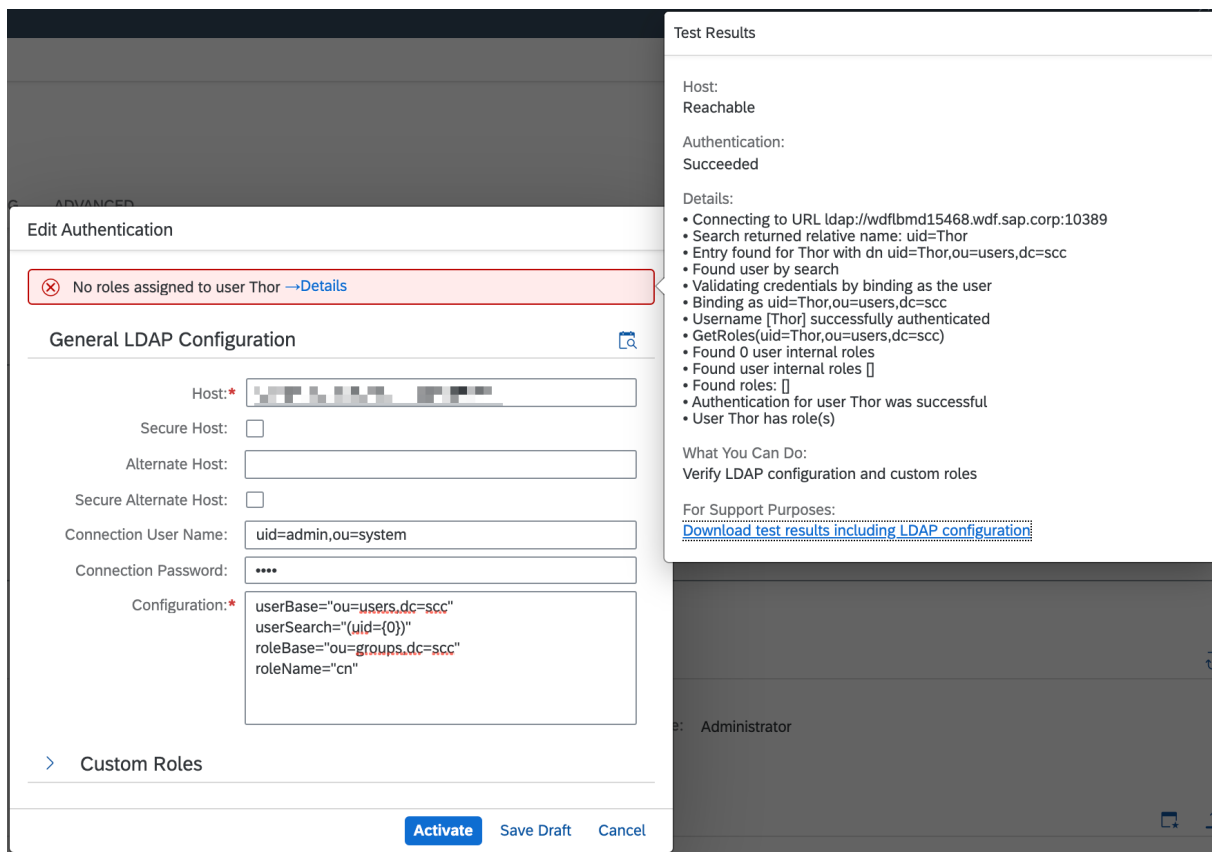
Like users, also groups are represented as nodes in an LDAP tree branch. The branch where the groups are located must be configured as `roleBase`.

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>" "(&(cn=<roleName>))"
```



When trying to define only `roleBase` and `roleName`, the test reports no roles for the specified user: *Found roles: []*. The empty brackets indicate an 'empty collection'. The reason is that the configuration does not define how users are related to the found groups. Find some background information on this in the next section.

Back to [Top \[page 525\]](#)

Relationship between User and Group

LDAP provides two ways to define the relationship between a user and its groups:

1. The group node contains one or more attributes `uniqueMember`, or
 2. Uses one or more attributes `memberOf` in a user node.
- **Case 1:** Extend the current configuration by `roleSearch="(uniqueMember={0})"`. The Cloud Connector's test reports direct roles in this case. Roles selected by `roleSearch` are LDAP entries. Using configuration parameter `roleName`, you specify the LDAP entry attribute which will be the role name. In our example, we are using `roleName="cn"` with the following result in the test: *"Found direct role cn=sccadmin,ou=groups,dc=scc -> sccadmin"*. The selected value `sccadmin` is the value of the attribute `cn` located in the original LDAP entry.

- **Case 2:** Extend the current configuration by `userRoleName="memberOf"`. This is reported by the Cloud Connector's test as internal role. If no groups are defined by `memberOf`, the internal group list is empty.

→ Remember

The roles specified by an attribute of a user entry are used as is, that is, the configuration parameter `roleName` is irrelevant in this case, and the role name is just set to the attribute value.

Below, the test report selected neither internal nor direct roles:

The screenshot displays the 'Edit Authentication' window for the Cloud Connector. The 'General LDAP Configuration' section is active, showing fields for Host, Secure Host, Alternate Host, Secure Alternate Host, Connection User Name, and Connection Password. The Configuration field is expanded, showing the following settings:

```

userBase="ou=users,dc=scc"
userSearch="(uid={0})"
roleBase="ou=groups,dc=scc"
roleName="cn"
userRoleName="memberOf"
roleSearch="(uniqueMemberX={0})"

```

A red error message at the top of the configuration section states: "No roles assigned to user Thor →Details". To the right, the 'Test Results' panel shows the following details:

- Host: Reachable
- Authentication: Succeeded
- Details:
 - Connecting to URL ldap://wdfbmd15468.wdf.sap.corp:10389
 - Search returned relative name: uid=Thor
 - Entry found for Thor with dn uid=Thor,ou=users,dc=scc
 - 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](#)

At the bottom of the configuration window, there are buttons for 'Activate', 'Save Draft', and 'Cancel'. The 'Custom Roles' section is also visible below the configuration fields.

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`:

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"`.

Custom User Roles

Note

Keep in mind that the custom role definition replaces the standard role. So, once a custom role for *Administrator* is set, the standard one (`sccadmin`) is not effective anymore.

SAP BTP Connectivity for the Neo Environment

Additional Notes

There are some more configuration parameters available that are out of scope here. Most LDAP configurations are covered by the parameters discussed above.

For special purposes, you may have to add to your configuration:

- `adCompat="true"`, if your LDAP server uses *MS Active Directory* and you encounter strange errors in the test report.
- `forceDnHexEscape="true"`, if your LDAP server uses *MS Active Directory* and there are non-standard characters in the DN.
- `connectionTimeout="x"`, if you want to change the default of 5s.

General Recommendations

- Don't use the `userPattern` parameter. It invalidates SSL/TLS-based authentication and high availability setup would fail.
- If the user ID has **non-standard characters**, escape them with `\nn`.
- For **back-slash**, always use `\\`.

Back to [Top \[page 525\]](#)

1.5.3.3 High Availability Setup

You can operate the Cloud Connector in a high availability mode, in which a master and a shadow instance are installed.

Task	Description
Install a Failover Instance for High Availability [page 538]	Install a redundant Cloud Connector instance (shadow) that monitors the main instance (master).
Master and Shadow Administration [page 541]	Learn how to operate master and shadow instances.

Related Information

[Connect DB Tools to SAP HANA via Service Channels \[page 495\]](#)

1.5.3.3.1 Install a Failover Instance for High Availability

The Cloud Connector lets you install a redundant instance that monitors the main instance.

Context

In a failover setup, when the main instance should go down for some reason, a redundant one can take over its role. The main instance of the Cloud Connector is called **master** and the redundant instance is called the **shadow**. The shadow has to be installed and connected to its master. During the setup of high availability, the master pushes the entire configuration to the shadow. Later on, during normal operation, the master also pushes configuration updates to the shadow. Thus, the shadow instance is kept synchronized with the master instance. The shadow pings the master regularly. If the master is not reachable for a while, the shadow tries to take over the master role and to establish the tunnel to SAP BTP.

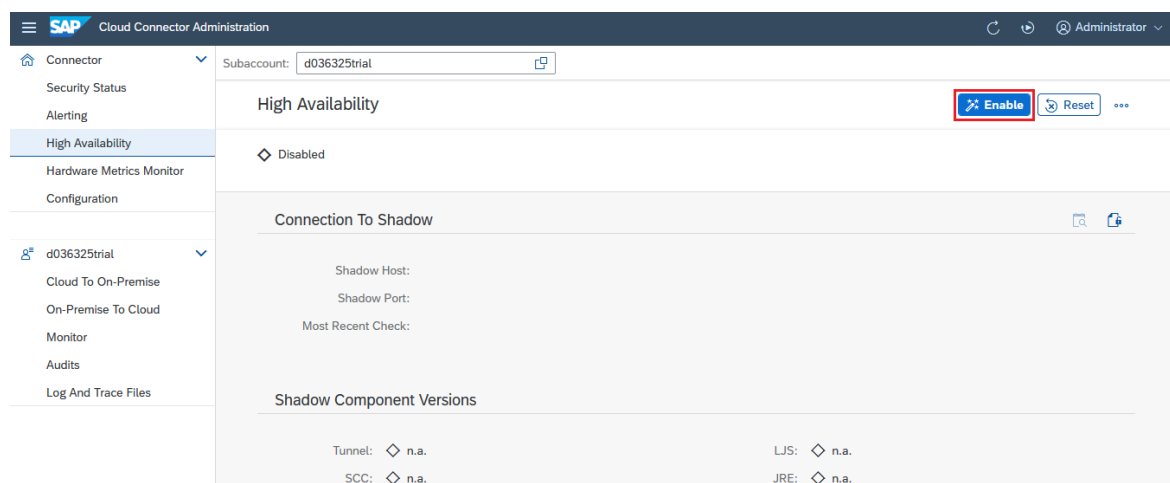
Note

For detailed information about sizing of the master and the shadow instance, see also [Sizing Recommendations \[page 258\]](#).

Procedure

Preparing the Master Instance for High Availability

1. Open the Cloud Connector UI and go to the master instance.
2. From the main menu, choose [High Availability](#).
3. Choose [Enable](#).



If this flag is not activated, no shadow instance can connect to this Cloud Connector. Additionally, when providing a concrete [Shadow Host](#), you can ensure that only from this host a shadow instance can be connected.

⚠ Caution

Pressing the [Reset](#) button resets all high availability settings to their initial state. As a result, high availability is disabled and the shadow host is cleared. Reset only works if no shadow is connected.

Installing and Setting Up a Shadow Instance

Install the shadow instance in the same network segment as the master instance. Communication between master and shadow via proxy is not supported. The same distribution package is used for master and shadow instance.

📘 Note

If you plan to use LDAP for the user authentication on both master and shadow, make sure you configure it **before** you establish the connection from shadow to master.

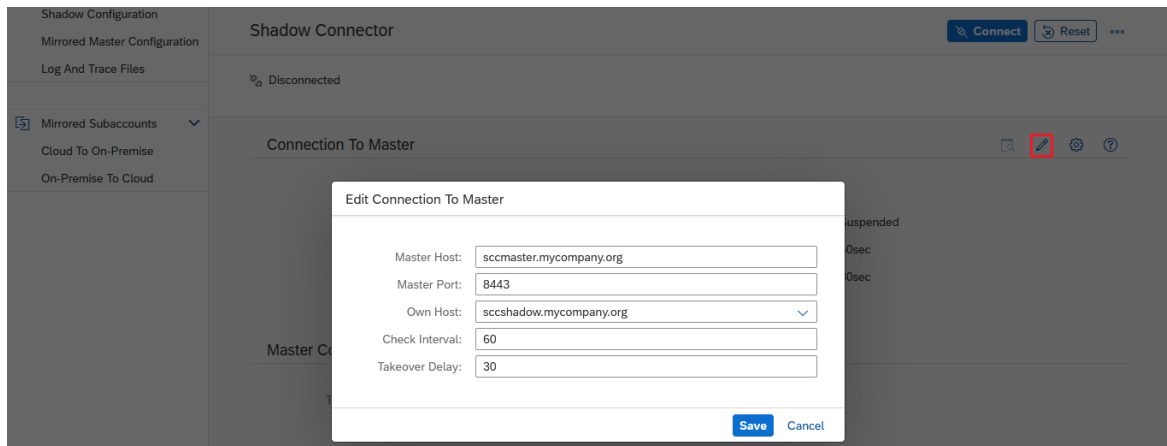
1. On first start-up of a Cloud Connector instance, a UI wizard asks you whether the current instance should be master or shadow. Choose [Shadow](#) and [Save](#):

 **Save**

Choose Installation Type

- ☐ Master (Primary Installation)
- ☒ Shadow (Backup Installation)

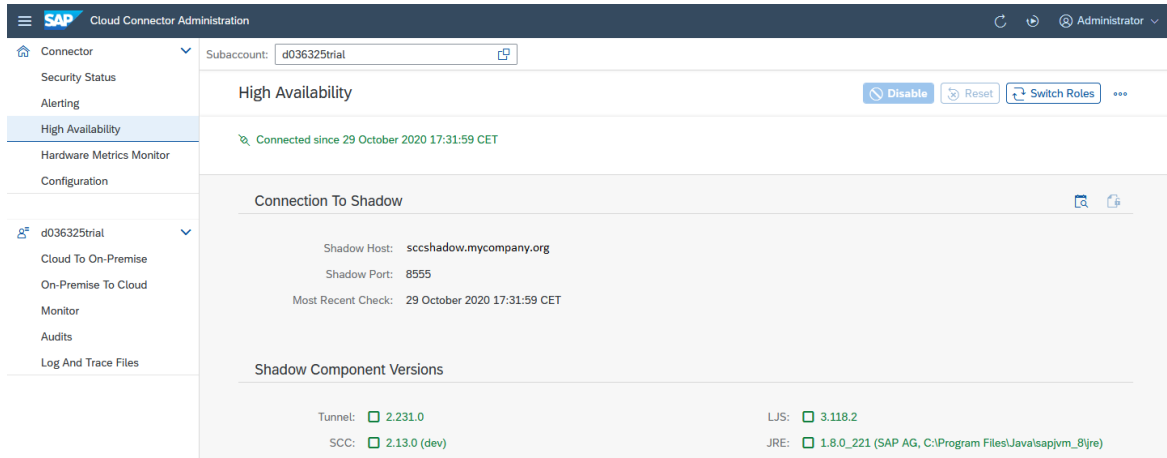
2. From the main menu, choose [Shadow Connector](#) and provide connection data for the master instance, that is, the master host and port. As of version 2.8.1.1, you can choose from the list of known host names, to use the host name under which the shadow host is visible to the master. You can specify 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.



Note

If you want to attach the shadow instance to a different master, press the [Reset](#) button. All your high availability settings will be removed, that is, reset to their initial state. This works only if the shadow is not connected.

- Upon a successful connection, the master instance pushes the entire configuration plus some information about itself to the shadow instance. You can see this information in the UI of the shadow instance, but you can't modify it.
- The UI on the master instance shows information about the connected shadow instance. From the main menu, choose [High Availability](#):



- 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

[Initial Configuration \[page 278\]](#)

[Master and Shadow Administration \[page 541\]](#)


1.5.3.3.2 Master and Shadow Administration

Administration of Shadow Instances

There are several administration activities you can perform on the shadow instance. All configuration of tunnel connections, host mappings, access rules, and so on, must be maintained on the master instance; however, you can replicate them to the shadow instance for display purposes. You may want to modify the **check interval** (time between checks of whether the master is still alive) and the **takeover delay** (time the shadow waits to see whether the master would come back online, before taking over the master role itself).

As of Cloud Connector version 2.11.2, you can configure the timeout for the connection check, by pressing the gear icon in the section [Connection To Master](#) of the shadow connector main page.

Edit Timeout Settings

 Validate and/or change current timeout settings

Timeout Settings

Connection Timeout (in ms):

1000

Request Timeout (in ms):

2000

Timeout Validation

Number of requests:

10

Requests:

Errors:

Connection Timeouts:

Request Timeouts:

Save

Cancel

SAP BTP Connectivity for the Neo Environment
Connectivity for the Neo Environment

PUBLIC

541

The [<Connection Timeout>](#) field specifies the maximum time allowed for establishing the technical connection to the master, the [<Request Timeout>](#) defines the maximum time allowed for executing the check over this connection. In the [Timeout Validation](#) section, you can check the current settings by pressing the [Execute](#) button. If the timeouts are hit, you might want to increase the settings accordingly.

Keep in mind the following points:

- The log level on master and shadow instances can be different.
- Configuration for check interval and takeover delay is maintained only on the shadow instance, and is transferred to the master for display purposes.
- Audit logs are only written on the master instance and are not transferred to the shadow. However, if the shadow becomes the master for some time, the audit log is potentially distributed over both master and shadow instances.

You can use the [Reset](#) button to drop all the configuration information on the shadow that is related to the master, but only if the shadow is not connected to the master.

Required Configuration on the Shadow Instance

Once connected to the master, the shadow instance receives the configuration from the master instance. Yet, there are some aspects you must configure on the shadow instance separately:

- **User administration** is configured separately on master and shadow instances. Generally, it is not required to have the same configuration on both instances. In most cases, however, it is suitable to configure master and shadow in the same way.
- The **UI certificate** is not shared. Each host can have its own certificate, so you must maintain the UI certificates on master and shadow. You can use the same certificate though.
- **SNC configuration**: If secure RFC communication or principal propagation for RFC calls is used, you must configure SNC on each instance separately.

Failover Process

The shadow instance regularly checks whether the master instance is still alive. If a check fails, the shadow instance first attempts to reestablish the connection to the master instance for the time period specified by the takeover delay parameter.

- If no connection becomes possible during the takeover delay time period, the shadow tries to take over the master role. At this point, it is still possible for the master to be alive and the trouble to be caused by a network issue between the shadow and master.
The shadow instance next attempts to establish a tunnel to the given SAP BTP subaccount. If the connection attempt fails to all configured subaccounts (for whatever reason), the shadow instance remains in "shadow status", periodically pinging the master and trying to connect to the cloud, while the master is not yet reachable.
- Otherwise, if the tunnel to the cloud side can be opened, the shadow instance will take over the master role. From this moment, the shadow instance displays the UI of a master instance and allows the usual

operations of a master instance, for example, starting/stopping tunnels, modifying the configuration, and so on.

This is, in particular, also the case if the master is still alive, but the network connection between shadow and master is corrupted. This leads to a *master-master* setup, which is automatically detected on the former master once the network between the two instances recovers. The former master will then automatically relinquish its master role and assume the shadow role to get back to the desired setup.

When the original master instance restarts, it first checks whether the registered shadow instance has taken over the master role. If it has, the master registers itself as a shadow instance on the former shadow (now master) instance. Thus, the two Cloud Connector installations, in fact, have switched their roles.

📘 Note

Only one shadow instance is supported. Any further shadow instances that attempt to connect are declined by the master instance.

The master considers a shadow as lost, if no check/ping is received from that shadow instance during a time interval that is equal to three times the check period. Only after this much time has elapsed can another shadow system register itself.

📘 Note

On the master, you can manually trigger failover by selecting the [Switch Roles](#) button. If the shadow is available, the switch is made as expected. Even if the shadow instance cannot be reached, the role switch of the master may still be enforced. Select [Switch Roles](#) only if you are absolutely certain it is the correct action to take for your current circumstances.

Even with the active role switch, zero downtime is not guaranteed. Depending on various aspects and timings, there may be short time slots in which establishing new connections fails. When switching the role, all active requests on the master will be broken as the sockets will be closed.

1.5.3.4 Change the UI Port

Context

By default, the Cloud Connector uses port **8443** for its administration UI. If this port is blocked by another process, or if you want to change it after the installation, you can use the `changeport` tool, provided with Cloud Connector version **2.6.0** and higher.

📘 Note

On Windows, you can also choose a different port during installation.

Procedure

1. Change to the installation directory of the Cloud Connector. To adjust the port and execute one of the following commands:

- Microsoft Windows OS:

```
changeport <desired_port>
```

- Linux OS, Mac OS X:

```
./changeport.sh <desired_port>
```

2. When you see a message stating that the port has been successfully modified, restart the Cloud Connector to activate the new port.

1.5.3.5 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.

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

Note

Once the Cloud Connector is freshly installed and connected to a cloud subaccount, none of the systems in the customer network are yet accessible to the applications of the related cloud subaccount. Accessible systems and resources must be configured explicitly in the Cloud Connector one by one, see [Configure Access Control \[page 340\]](#).

A Cloud Connector instance can be connected to multiple subaccounts in the cloud. This is useful especially if you need multiple subaccounts to structure your development or to stage your cloud landscape into

development, test, and production. In this case, you can use a single Cloud Connector instance for multiple subaccounts. However, we recommend that you do not use subaccounts running in productive scenarios and subaccounts used for development or test purposes within the same Cloud Connector. You can add or delete a cloud account to or from a Cloud Connector using the [Add](#) and [Delete](#) buttons on the [Subaccount Dashboard](#) (see screenshot above).

Related Information

[Managing Subaccounts \[page 291\]](#)

1.5.3.6 Secure the Activation of Traffic Traces

For support purposes, you can trace HTTP and RFC network traffic that passes through the Cloud Connector.

Context

Traffic data may include business-critical information or security-sensitive data, such as user names, passwords, address data, credit card numbers, and so on. Thus, by activating the corresponding trace level, a Cloud Connector administrator might see data that he or she is not meant to. To prevent this behavior, implement the four-eyes principle for your operating system as described below.

Once the four-eyes principle is applied, activating a trace level that dumps traffic data will require two separate users:

- An operating system user on the machine where the Cloud Connector is installed;
- An `Administrator` user of the Cloud Connector user interface.

By assigning these roles to two different people, you can ensure that both persons are needed to activate a traffic dump.

Four-Eyes Principle for Microsoft Windows OS

1. Create a file named `writeHexDump` in `<scc_install_dir>\scc_config`. The owner of this file must be a user other than the operating system user who runs the `cloud connector` process.

Note

Usually, this file owner is the user which is specified in the [Log On](#) tab in the properties of the `cloud connector` service (in the Windows [Services](#) console). We recommend that you do not use the *Local System* user, but a dedicated OS user for the `cloud connector` service.

- Only the file owner should have write permission for the file.
 - The OS user who runs the `cloud_connector` process needs read-only permissions for this file.
 - Initially, the file should contain a line like `allowed=false`.
 - In the security properties of the file `scc_config.ini` (same directory), make sure that only the OS user who runs the `cloud_connector` process has write/modify permissions for this file. The most efficient way to do this is simply by removing all other users from the list.
2. Once you've created this file, the Cloud Connector refuses any attempt to activate the *Payload Trace* or *SNC Payload Trace* flag.
 3. To set CPIC trace level 3 or activate any payload trace, first the owner of `writeHexDump` must change the file content from `allowed=false` to `allowed=true`. Thereafter, the *Administrator* user can activate any payload trace from the Cloud Connector administration screens.

Four-Eyes Principle for Linux OS/Mac OS X

1. Go to directory `/opt/sap/scc/scc_config` and create a file with name `writeHexDump`. The owner of this file must be different from the `scctunnel` user (that is, the operating system user under which the Cloud Connector processes are running) and not a member of the operating system user group `sccgroup`.
 - Only the file owner should have write permission for the file.
 - The `scctunnel` user needs read-only permissions for this file.
 - Initially, the file should contain a line like `allowed=false`.
2. Once you've created this file, the Cloud Connector refuses any attempt to activate the *Payload Trace* or *SNC Payload Trace* flag.
3. To set CPIC trace level 3 or activate any payload trace, first the owner of the `writeHexDump` file mentioned above must change the file content from `allowed=false` to `allowed=true`. Then, the *Administrator* user can activate any payload trace from the Cloud Connector administration screens.

1.5.3.7 Monitoring

Learn how to monitor the Cloud Connector from the SAP BTP cockpit and from the Cloud Connector administration UI.

Checking the Operational State

The simplest way to verify whether a Cloud Connector is running is to try to access its administration UI. If you can open the UI in a Web browser, the `cloud_connector` process is running.

- On Microsoft Windows operating systems, the `cloud_connector` process is registered as a Windows service, which is configured to start automatically after a new Cloud Connector installation. If the Cloud Connector server is rebooted, the `cloud_connector` process should also auto-restart immediately. You can check the state with the following command:

```
sc query "SAP Cloud Connector"
```

The line state shows the state of the service.

- On Linux operating systems, the Cloud Connector is registered as a daemon process and restarts automatically each time the `cloud_connector` process is down, for example, following a system restart. You can check the daemon state with the following command:

```
service scc_daemon status
```

To verify if a Cloud Connector is connected to a certain cloud subaccount, log on to the Cloud Connector administration UI and go to the [Subaccount Dashboard](#), where the connection state of the connected subaccounts is visible, as described in section [Connect and Disconnect a Cloud Subaccount](#) [page 544].

Monitoring from the Cockpit

The cockpit includes a [Connectivity](#) section, where users can check the status of the Cloud Connector(s) attached in the current subaccount, if any, as well as information about the Cloud Connector ID, version, used Java runtime, high availability setup (master and shadow instance), and so on (choose ► [Connectivity](#) ► [Cloud Connectors](#) ►).

Access to this view is granted to:

- Neo environment: Users with a role containing the permission `readSCCTunnels`, for example, the predefined role `Cloud Connector Admin`.
- Cloud Foundry environment, feature set A: Users with a Cloud Foundry org role containing the permission `readSCCTunnels`, for example, the role `Org Manager`.

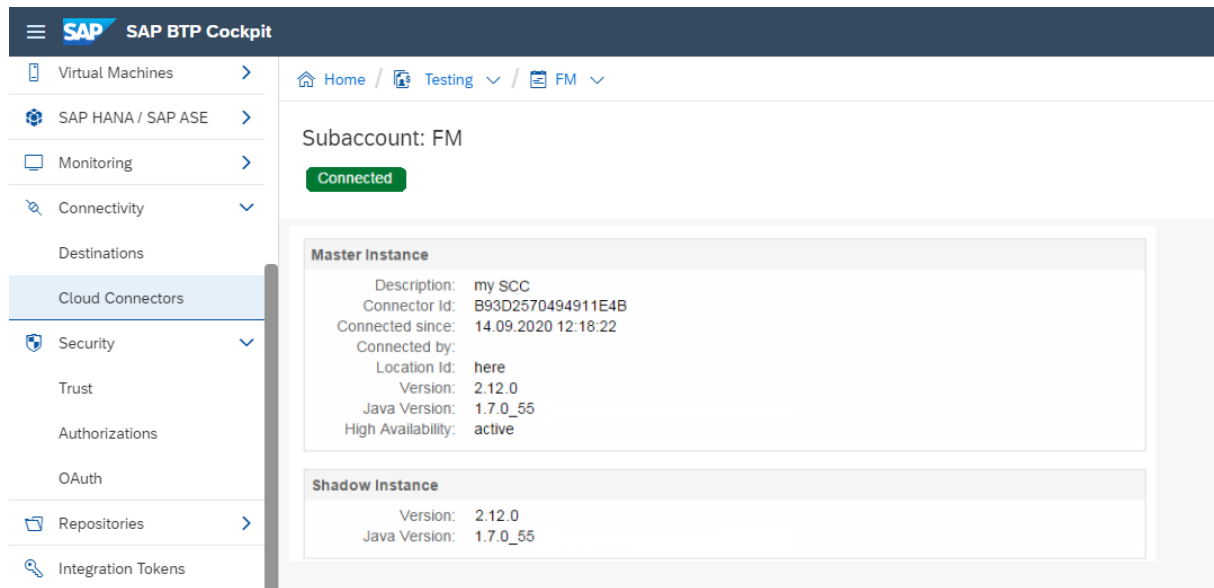
📘 Note

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

- Cloud Foundry environment, feature set B: Users with a role containing the permission `readSCCTunnels`, for example, the predefined role `Cloud Connector Administrator`.

📘 Note

For more information on feature sets in the Cloud Foundry environment, see [Cloud Management Tools — Feature Set Overview](#).



Monitoring from the Cloud Connector Administration UI

The Cloud Connector offers various views for monitoring its activities and state.

You can check the overall state of the Cloud Connector through its [Hardware Metrics \[page 548\]](#), whereas subaccount-specific performance and usage data is available via [Subaccount-Specific Monitoring \[page 550\]](#). To provide external monitoring tools, you can use the [Monitoring APIs \[page 556\]](#).

Related Information

[Configure Solution Management Integration \[page 506\]](#)

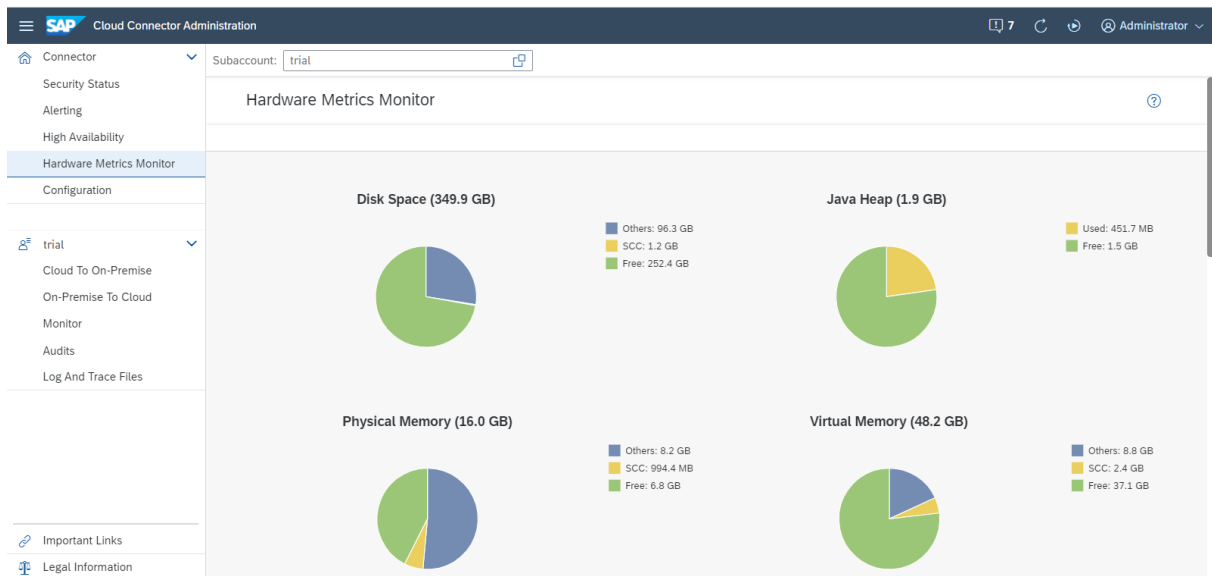
[High Availability Setup \[page 537\]](#)

1.5.3.7.1 Hardware Metrics

Check the current state of critical system resources in the Cloud Connector.

You can check the current state of critical system resources (disc space, Java heap, physical memory, virtual memory) using pie charts.

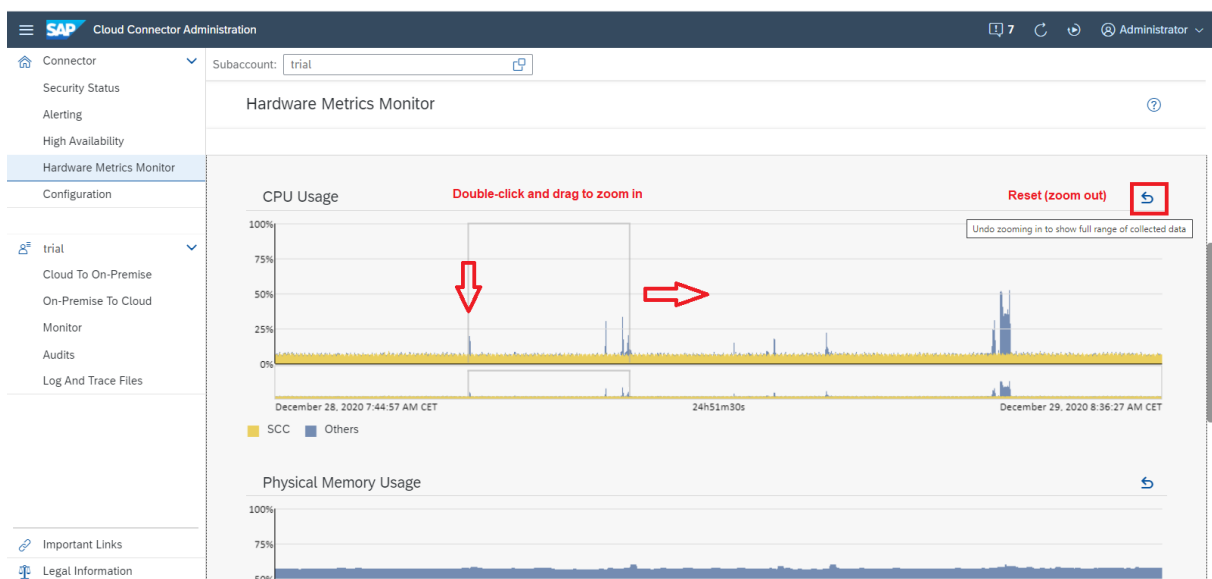
To access the monitor, choose [Hardware Metrics Monitor](#) from the main menu.



In addition, the history of CPU and memory usage (physical memory, Java heap) is shown in history graphs below the pie charts (recorded in intervals of 15 seconds), as well as the history of disk usage, recorded in intervals of 60 seconds. History data is retained for at most 24 hours.

You can view the usage data for a selected time period in each history graph:

- Double-click inside the main graph area to set the start (or end) point, and drag to the left or to the right to zoom in.
 - The entire timeline is always visible in the smaller bottom area right below the main graph.
 - A frame in the bottom area shows the position of the selected section in the overall timeline.
- Choose [Undo zooming in...](#) to reset the main graph area to the full range of available data.



Note

Zooming, dragging, and undoing zooming is synchronized across all history graphs. All graphs always show the situation during the same time period.

History data is written to a file to avoid data loss when stopping the Cloud Connector. Upon restart, the history data is read from file. Downtime is represented by a gray rectangle indicating that no data is available during the respective period of time.

1.5.3.7.2 Subaccount-Specific Monitoring

Use different monitoring views in the Cloud Connector administration UI to check subaccount-specific activities and data.

The Cloud Connector provides various views for monitoring the activities associated with an account, such as HTTP requests and RFC calls (from cloud applications to backends as per access control settings) or data statistics for service channels. Choose one of the sub-menus [Monitor \(Cloud to On-Premise\)](#) or [Monitor \(On-Premise to Cloud\)](#).

Caution

The collected monitoring data is not part of the Cloud Connector's backup. After restoring a Cloud Connector instance, all monitoring collections are empty.

[Monitoring \(Cloud to On-Premise\) \[page 550\]](#)

[Monitoring \(On-Premise to Cloud\) \[page 555\]](#)

1.5.3.7.2.1 Monitoring (Cloud to On-Premise)

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

Content

[Performance Overview \[page 551\]](#)

[Most Recent Requests \[page 551\]](#)

[Resource Filter Settings \[page 552\]](#)

[Top Time Consumers \[page 553\]](#)

[Usage Statistics \[page 554\]](#)

[Backend Connections \[page 555\]](#)

Performance Overview

All requests that travel through the Cloud Connector to a backend system, as specified through access control, take a certain amount of time. You can check the duration of requests in a bar chart. The requests are not shown individually, but are assigned to buckets, each of which represents a time range.

For example, the first bucket contains all requests that took 10ms or less, the second one the requests that took longer than 10ms, but not longer than 20ms. The last bucket contains all requests that took longer than 5000ms.

In case of latency gaps, you may try to adjust the influencing parameters: number of connections, tunnel worker threads, and protocol processor worker threads. For more information, see [Configuration Setup \[page 262\]](#).

The collection of duration statistics starts as soon as the Cloud Connector is operational. You can delete all of these statistical records by selecting the button *Delete All*. After that, the collection of duration statistics starts over.

Note

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

Back to [Content \[page 550\]](#)

Most Recent Requests

This option shows the most recent requests:

The screenshot shows the SAP Cloud Connector Administration interface. The left sidebar contains navigation options: Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, trial, Cloud To On-Premise, On-Premise To Cloud, Monitor, Audits, and Log And Trace Files. The main area is titled 'Monitor' and shows performance statistics collected since 21. Dezember 2020 15:14:26 MEZ. The 'PERFORMANCE' tab is active, and 'MOST RECENT REQUESTS' is selected. Below this, the 'Resource Filter Settings' table is visible, showing a filter for 'sales-system.cloud:443'. The 'Select Host' dropdown is set to 'All Hosts'. The main table displays the following data:

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

The number of requests that are shown is limited to 50. You can either view all requests or only the ones destined for a certain virtual host, which you can select. You can select a row to see more detail.



A horizontal stacked bar chart breaks down the duration of the request into several parts: external (backend), open connection, internal (Cloud Connector), SSO handling, and latency effects (between SAP BTP and Cloud Connector). The numbers in each part represent milliseconds.

Note

Sections with a duration of less than 1ms are not included.

In the above example, the selected request took 34ms, to which the Cloud Connector contributed 1ms. Opening a connection took 18ms. Backend processing consumed 7ms. Latency effects accounted for the remaining 8ms, while there was no SSO handling necessary and hence it took no time at all.

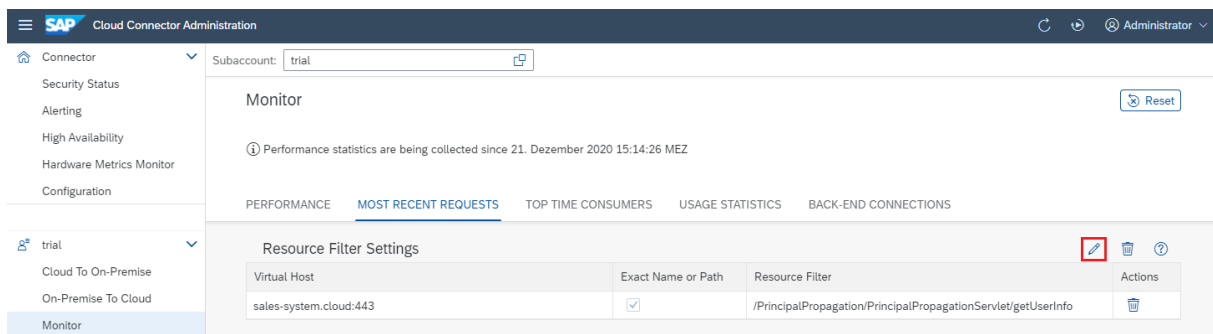
Note

The term "request" only refers to RFC and HTTP requests. TCP or LDAP traffic does not contribute to *Most Recent Requests* or *Top Time Consumers*.

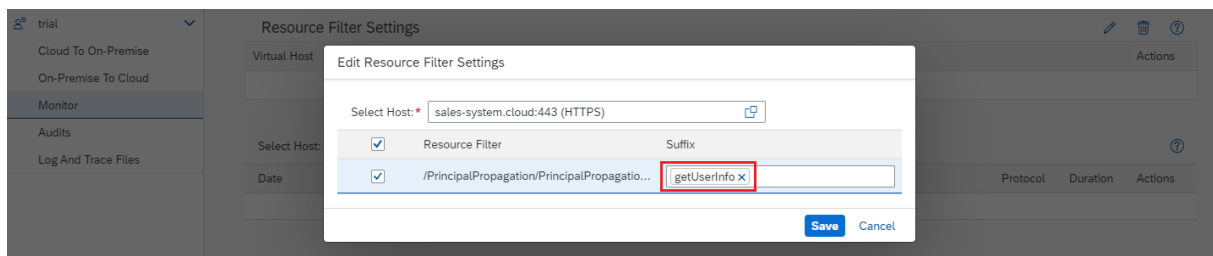
Back to [Content \[page 550\]](#)

Resource Filter Settings

To further restrict the selection of the listed 50 most recent requests, you can edit the resource filter settings for each virtual host:



In the [Edit](#) dialog, select the virtual host for which you want to specify the resource filter and choose one or more of the listed accessible resources. This list includes all resources that have been exposed during access control configuration (see also: [Configure Access Control \[page 340\]](#)). If the access policy for an accessible resource is set to `Path` and `all sub-paths`, you can further narrow the selection by adding one or more sub-paths to the resource as a suffix .



Each selected resource/sub-path is listed separately in the resource filter list.

Note

If you specify sub-paths for a resource, the request URL must match exactly one of these entries to be recorded. Without specified sub-paths (and the value `Path` and `all sub-paths` set for a resource), all sub-paths of a specified resource are recorded.

Back to [Content \[page 550\]](#)

Top Time Consumers

This option is similar to [Most Recent Requests](#); however, requests are not shown in order of appearance, but rather sorted by their duration (in descending order). Furthermore, you can delete top time consumers, which has no effect on most recent requests or the performance overview.

Back to [Content \[page 550\]](#)

Usage Statistics

To view the statistical data regarding the traffic handled by each **virtual host**, you can select a virtual host from the table. The detail view shows the traffic handled by *each resource*, as well as a *24 hour overview* of the throughput as a bar chart that aggregates the throughput (bytes received and bytes sent by a virtual host, respectively) on an hourly basis.

Note

For communication via HTTP and RFC, also the number of calls or requests is recorded in the same way.

The screenshot shows the SAP Cloud Connector Administration web interface. The left sidebar contains a navigation menu with options like Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, trial, Cloud To On-Premise, On-Premise To Cloud, Monitor, Audits, and Log And Trace Files. The main area is titled 'Monitor' and shows usage statistics for a subaccount named 'trial'. It includes tabs for PERFORMANCE, MOST RECENT REQUESTS, TOP TIME CONSUMERS, USAGE STATISTICS (selected), and BACK-END CONNECTIONS. Under 'Virtual Systems Usage (1)', there is a table with columns: Prot..., Virtual Host, Bytes Received, Bytes Sent, Most Recent Access, Calls, and Actions. The table shows one entry for 'RFC' with 21.4 KB received, 12.8 KB sent, and 20 calls. Below this, 'Resources Usage Of' shows a table with columns: Status, Function Name, Bytes Received, Bytes Sent, Most Recent Access, Calls, and Actions. It shows one entry for 'STFC_CONNECTION' with 6190 Bytes received and 4803 Bytes sent. At the bottom, there is a '24H Throughput Overview' section with a bar chart titled 'Bytes Received' showing data for December 26 and 27, 2020.

The data that is collected includes the number of bytes received *from* cloud applications and the number of bytes sent back to cloud applications. The time of the most recent access is shown for the virtual hosts, and in the detail view also for the resources. If no access has taken place yet, the most recent access is shown as n . a . (not available). Similarly, the number of bytes received and sent of a virtual host is the sum of bytes received and sent of its resources.

The tables listing usage statistics of virtual hosts and their resources let you delete unused virtual hosts or unused resources. Use action [Delete](#) to delete such a virtual host or resource.

Caution

In Cloud Connector versions **before 2.14**, usage statistics are collected during runtime only and are not stored when stopping the Cloud Connector. That is, these statistics are lost when the Cloud Connector is stopped or restarted. Use care when taking the decision to delete a resource or virtual host based on its usage statistics.

As of version 2.14, usage statistics are periodically stored to disk. The collected statistics are still available after a restart of the Cloud Connector.

Actively deleted statistics are gone in either case.

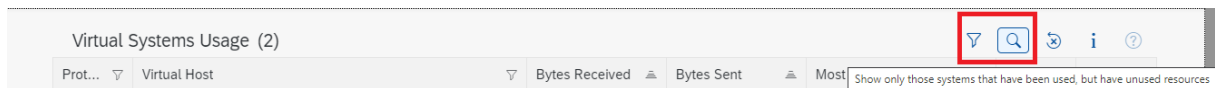
A backup of the Cloud Connector settings does not include collected usage data. After restoring a Cloud Connector instance, all monitoring collections are empty.

Using the [Reset](#) button, you can clean up all collected data.

⚠ Caution

Statistical data will be deleted permanently.

For both virtual hosts and resources, you can use a classic [Filter](#) button to reduce the virtual hosts or resources to those that have never been used (since the Cloud Connector started). For the virtual hosts, a second filter type is available that selects only those virtual hosts that have been used, but *include resources never used*. This feature facilitates locating obsolete resources of otherwise active virtual hosts.



Back to [Content \[page 550\]](#)

Backend Connections

This option shows a tabular overview of all active and idle connections, aggregated for each virtual host. By selecting a row (each of which represents a virtual host) you can view the details of all active connections as well as a graphical summary of all idle connections. The graphical summary is an accumulative view of connections based on the time the connections have been idle.

The maximum idle time appears on the rightmost side of the horizontal axis. For any point t on that axis (representing a time value ranging between 0ms and the maximal idle time), the ordinate is the number of connections that have been idle for no longer than t . You can click inside the graph area to view the respective abscissa t and ordinate.

Back to [Content \[page 550\]](#)

1.5.3.7.2.2 Monitoring (On-Premise to Cloud)

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

Connections

This section shows a tabular overview of all currently opened logical connections, aggregated for each local port. You can identify if and how many connections are currently opened through this specific service channel.

Usage Statistics

Statistical data regarding the traffic handled by each port is shown in tabular form. From the table, you can select a service channel. The respective detail views show a 24 hour overview as a bar chart that aggregates the throughput (bytes received and bytes sent via a port, respectively) on an hourly basis.

The collected data comprises the number of bytes received from on-premise applications and the number of bytes sent to cloud applications.

The table listing usage statistics of ports lets you delete unused service channels. Use action [Delete](#) to delete a service channel.

⚠ Caution

Usage statistics are collected at runtime only. They are not stored when stopping the Cloud Connector. These statistics are lost when the Cloud Connector is stopped or restarted. Be mindful of that fact, and use caution when taking the decision to delete a service channel based on its usage statistics.

Using the [Reset](#) button you can clean up all collected data. This might be helpful when monitoring a specific use case.

The table of service channels provides a filter button to reduce the service channels to those that have never been used (since the Cloud Connector started).

1.5.3.7.3 Monitoring APIs

Use the Cloud Connector monitoring APIs to include monitoring information in your own monitoring tool.

Context

You might want to integrate some monitoring information in the monitoring tool you use.

For this purpose, the Cloud Connector includes a collection of APIs that allow you to read various types of monitoring data.

📘 Note

This API set is designed particularly for monitoring the Cloud Connector via the SAP Solution Manager, see [Configure Solution Management Integration \[page 506\]](#).

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`.

Note

The Health Check API does not require a specified user. Separate users are available through LDAP only.

Available APIs

The following APIs are currently available.

- [Health Check \[page 557\]](#) (available as of version 2.16.0)
- [Subaccount Data \[page 558\]](#) (as of 2.10.0)
- [Open Connections to On-Premise Backend Systems \[page 559\]](#) (as of 2.10.0)
- [Open Connections to Cloud Services \[page 560\]](#) (as of 2.15.0)
- [Performance Data \[page 561\]](#) (as of 2.10.0)
- [Top Time Consumers \[page 563\]](#) (as of 2.11.0)
- [Memory Status \[page 565\]](#) (as of 2.13.0)
- [Certificate Status \[page 567\]](#) (as of 2.13.0)
- [Certificate Selection List \[page 569\]](#) (as of 2.13.0)
- [Usage Statistics \[page 570\]](#) (as of 2.13.0)
- [Master Role Check \[page 573\]](#) (as of 2.15.0)

Health Check (available as of version 2.16.0)

Using the health check API, it is possible to recognize that the Cloud Connector is up and running. The purpose of this health check is only to verify that the Cloud Connector is not down. It does not check any internal state or tunnel connection states. Thus, it is a quick check that you can execute frequently:

URI	<code>/exposed?action=ping</code>
Method	<code>GET</code>
Request	
Response	
Errors	
Roles	All roles are accepted

Back to [Available APIs \[page 557\]](#)

Back to [Context \[page 556\]](#)

List of Subaccounts (available as of version 2.10.0)

Note

This API is relevant for the master instance only.

Using this API, you can read the list of all subaccounts connected to the Cloud Connector and view detail information for each subaccount:

URI	/api/monitoring/subaccounts
Method	GET
Request	
Response	
Errors	
Roles	Administrator, Monitoring

Response Properties:

- **subaccounts:** array of subaccounts for which data is provided
 - **regionHost:** host of the region, in which the subaccount is residing
 - **subaccount:** name of subaccount
 - **displayName:** display name of the subaccount
 - **description:** description for the subaccount
 - **locationID:** identifying the location of this Cloud Connector for a specific subaccount
 - **tunnel:** array of connection tunnels used by the subaccount
 - **state:** *Connected*, *ConnectFailure*, or *Disconnected*
 - **connectedSinceTimeStamp:** connection start time as UTC timestamp
 - **connections:** number of subaccount connections
 - **applicationConnections:** array of connections to application instances
 - **serviceChannels:** type and state of the service channels used (types: HANA database, Virtual Machine or RFC)
 - **subaccountCertificate:** information on the subaccount certificate such as validity period, issuer and subject DN
- **version:** API version.

Example:

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

Back to [Available APIs \[page 557\]](#)

Back to [Context \[page 556\]](#)

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

Note

This API is relevant for the master instance only.

The list of connections lets you view all backend systems connected to the Cloud Connector and get detail information for each connection:

URI	<code>/api/monitoring/connections/backends</code>
Method	GET
Request	
Response	<code>{subaccounts, version}</code>
Errors	
Roles	Administrator, Monitoring

Response Properties:

- `subaccounts`: array of subaccounts for which data is provided
 - `regionHost`: host of the region, in which the subaccount is residing
 - `subaccount`: name of subaccount
 - `locationID`: identifying the location of this Cloud Connector for a specific subaccount
 - `backendConnections`: array of connections to a specified backend system
 - `virtualBackend`: virtual (external) backend URL
 - `internalBackend`: internal backend URL
 - `protocol`: type of protocol (RFC, HTTP, and so on)
 - `idle`: number of idle connections
 - `active`: number of active connections
- `version`: API version.

Sample Code

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

Example:

```

{
  "subaccounts": [
    {
      "backendConnections": [],
      "regionHost": "hana.ondemand.com",
      "subaccount": "a117",
      "locationID": ""
    },
    {
      "backendConnections": [
        {
          "virtualBackend": "abapserver.hana.cloud:",
          "internalBackend": "sap.corp:",
          "protocol": "RFC",
          "idle": 1,
          "active": 0
        }
      ],
      "regionHost": "hana.ondemand.com",
      "subaccount": "dev",
      "locationID": ""
    },
    {
      "backendConnections": [],
      "regionHost": "hanatrial.ondemand.com",
      "subaccount": "trial",
      "locationID": ""
    }
  ],
  "version": 1
}

```

Back to [Available APIs \[page 557\]](#)

Back to [Context \[page 556\]](#)

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

Note

This API is relevant for the master instance only.

The list of connections opened for service channels:

URI	/api/monitoring/connections/serviceChannels
Method	GET

Request	
Response	{subaccounts, version}
Errors	
Roles	Administrator, Monitoring

Response Properties:

- **subaccounts:** array of subaccounts for which data is provided
 - **regionHost:** host of the region, in which the subaccount is residing
 - **subaccount:** name of subaccount
 - **locationID:** identifying the location of this Cloud Connector for a specific subaccount
 - **serviceChannelConnections:** array of connections opened by the specified service channel
 - **port:** port of the service channel
 - **typeDesc:**
 - **typeKey:** key name of the service channel, for example, *ABAPCloud*
 - **typeName:** human readable service channel type
 - **connections:** number of connections
- **version:** API version.

Sample Code

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

Back to [Available APIs \[page 557\]](#)

Back to [Context \[page 556\]](#)

Performance Monitor Data (available as of version 2.10.0)

Note

This API is relevant for the master instance only.

Using this API, you can read the data provided by the Cloud Connector performance monitor:

URI	/api/monitoring/performance/backends
Method	GET
Request	

Response	{subaccounts, version}
Errors	
Roles	Administrator, Monitoring

Response Properties:

- subaccounts: array of subaccounts for which data is provided
 - regionHost: host of the region, in which the subaccount is residing
 - subaccount: name of subaccount
 - locationID: identifying the location of this Cloud Connector for a specific subaccount
 - backendPerformance given as array of:
 - virtualHost: host name of the backend system
 - virtualPort: port of the backend system
 - protocol: type of protocol (RFC, HTTP etc.)
 - buckets: array of performance data related to backend system
 - numberOfCalls: number of calls performed between Cloud Connector and backend system
 - minimumCallDurationsMs: minimum duration of the executed calls in milliseconds
 - sinceTime: start of performance measurement
- version: API version.

Sample Code

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

Example:

```

▼ {
  ▼ "subaccounts": [
    ▼ {
      "backendPerformance": [],
      "sinceTime": "2017-05-11T15:48:42.084 +0200",
      "regionHost": "hana.ondemand.com",
      "subaccount": "a117",
      "locationID": ""
    },
    ▼ {
      ▼ "backendPerformance": [
        ▼ {
          "virtualHost": "abapserver.hana.cloud",
          "virtualPort": "2",
          "protocol": "RFC",
          ▼ "buckets": [
            ▼ {
              "numberOfCalls": 0,
              "minimumCallDurationMs": 0
            },
            ▼ {
              "numberOfCalls": 1,
              "minimumCallDurationMs": 10
            },
            ▼ {
              "numberOfCalls": 0,
              "minimumCallDurationMs": 20
            },
            ▼ {
              "numberOfCalls": 0,
              "minimumCallDurationMs": 30
            }
          ]
        }
      ]
    }
  ]
}

```

Back to [Available APIs \[page 557\]](#)

Back to [Context \[page 556\]](#)

Top Time Consumers (available as of version 2.11.0)

📌 Note

This API is relevant for the master instance only.

Using this API, you can read the data of top time consumers provided by the Cloud Connector performance monitor:

URI	/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>:<scport>/api/monitoring/performance/topTimeConsumers
```

Example:


```

{
  "subaccounts": [
    {
      "sinceTime": "2017-06-26T16:57:21.615 +0200",
      "requests": [
        {
          "startTime": "2017-06-26T16:57:25.426 +0200",
          "id": 1639964397,
          "protocol": "RFC",
          "virtualBackend": "abapserver.hana.cloud:sapgw42",
          "internalBackend": "ldciv9u:sapgw51",
          "resource": "RFCPING",
          "sentBytes": 307,
          "receivedBytes": 441,
          "user": "JCUSER",
          "totalTime": 21,
          "externalTime": 6,
          "genSsoTime": 0,
          "openRemoteTime": 5,
          "validateSsoTime": 0,
          "latencyTime": 6
        },
        { ... } // 15 items
      ],
      "regionHost": "int.sap.hana.ondemand.com",
      "subaccount": "d039407sapdev",
      "locationID": "location"
    }
  ],
  "version": 1
}

```

Back to [Available APIs \[page 557\]](#)

Back to [Context \[page 556\]](#)

Memory Status (available as of version 2.13.0)

Note

This API is relevant for the master instance only.

This API provides a snapshot of the current memory status of the machine where the Cloud Connector is running:

URI	<code>/api/monitoring/memory</code>
Method	GET
Request	

Response	{physicalKB, virtualKB, cloudConnectorHeapKB}
Errors	
Roles	Administrator, Monitoring

Response Properties:

- `physicalKB`: usage of the physical memory, split into four categories (all sizes in KB):
 - `total`: 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>:<scport>/api/monitoring/memory
```

Example:

```

{
  "physicalKB": {
    "total": 33380516,
    "CloudConnector": 633932,
    "others": 10960464,
    "free": 21786120
  },
  "virtualKB": {
    "total": 35477668,
    "CloudConnector": 1267504,
    "others": 14174208,
    "free": 20035956
  },
  "cloudConnectorHeapKB": {
    "total": 983040,
    "used": 235457,
    "free": 747583
  }
}

```

Back to [Available APIs \[page 557\]](#)

Back to [Context \[page 556\]](#)

Certificate Status (available as of version 2.13.0)

Note

This API is relevant for the master instance only.

Using this API, you can get an overview of the certificates currently employed by the Cloud Connector:

URI	<code>/api/monitoring/certificates</code>
Method	GET
Request	
Response	<code>{expired, expiring, ok}</code>
Errors	
Roles	Administrator, Monitoring

Response Properties:

- `expired`: 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 the subaccount (only for subaccount certificates)

Sample Code

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

Example:

```
{
  "expired": [
    {
      "type": "System",
      "subjectDN": "CN\u003dHugo, OU\u003dCSI, O\u003dSAP Trust Community, C\u003dDE",
      "validTo": 1458290342000
    }
  ],
  "expiring": [],
  "ok": [
    {
      "type": "UI",
      "subjectDN": "CN\u003dSCC, OU\u003dConnectivity, O\u003dSAP SE, C\u003dDE",
      "validTo": 1791826434000
    },
    {
      "type": "subaccount",
      "subaccountName": "d036325trial",
      "subaccountRegion": "hanatrial.ondemand.com",
      "validTo": 1613132865000
    }
  ]
}
```

Back to [Available APIs \[page 557\]](#)

Back to [Context \[page 556\]](#)

Certificate Selection List (available as of version 2.13.0)

Note

This API is relevant for the master instance only.

Using this API, you can obtain an overview of the certificates currently employed by the Cloud Connector:

URI	<code>/api/monitoring/certificates/{selection}</code>
Method	GET
Request	
Response	[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
 - `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)
- `subaccountName`: the name of the subaccount (only for subaccount certificates)
- `subaccountRegion`: the region or landscape host of the the subaccount (only for subaccount certificates)

Sample Code

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

Back to [Available APIs \[page 557\]](#)

Back to [Context \[page 556\]](#)

Usage Statistics (available as of version 2.13.0)

Note

This API is relevant for the master instance only.

This API provides usage statistics regarding the systems and resources available in the Cloud Connector:

URI	<code>/api/monitoring/usage</code>
Method	GET
Request	
Response	<code>{subaccounts, version}</code>
Errors	
Roles	Administrator, Monitoring

Response Properties:

- `subaccounts`: array of subaccounts for which data is provided
 - `regionHost`: host of the region, in which the subaccount is residing
 - `subaccount`: name of subaccount
 - `locationID`: identifying the location of this Cloud Connector for a specific subaccount
 - `usageStatistics`: backend usage statistics, given as an array of
 - `virtualHost`: host name of the backend system
 - `virtualPort`: port of the backend system
 - `protocol`: type of protocol (RFC, HTTP etc.)
 - `bytesReceived`: total number of bytes that were received through a call or request
 - `bytesSent`: total number of bytes sent back as a response
 - `calls`: total number of calls or requests
 - `mostRecentAccess`: time of the most recent access (that is, call or request) given as a UTC timestamp;

Note

This property is only available if there has been at least one call or request.

- **resources:** usage statistics per resource, given as an array (i.e. the distribution of bytes received, sent, as well as number of calls/requests, across the resources of the respective virtual host)
 - **resourceName:** name of the resource (that is, a URL path or the name of a remote function)
 - **enabled:** Boolean flag that indicates whether the resource is currently active (true) or suspended (false)
 - **bytesReceived:** total number of bytes that were received through a call or request and were handled by this resource
 - **bytesSent:** total number of bytes sent back as a response in the context of this resource
 - **calls:** total number of calls or requests handled by this resource
 - **mostRecentAccess:** time of the most recent access (that is, call or request) given as a UTC timestamp;

Note

This property is only available if at least one call or request was handled by this resource.

- **serviceChannelUsageStatistics:** service channels usage statistics, given as an array of
 - **port:** port of the service channel
 - **typeDesc:**
 - **typeKey:** key name of the service channel, for example, *ABAPCloud*
 - **typeName:** human readable service channel type
 - **bytesReceived:** total number of bytes received through the service channel
 - **bytesSent:** total number of bytes sent back through the service channel
- **sinceTime:** start of performance measurement
- **version:** API version.

Sample Code

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

Example:

```
{
```

```
  "subaccounts": [  
    {  
      "subaccountName": "lannister",  
      "subaccountRegion": "westeros.com",  
      "sinceTime": 1596810677689,  
      "usageStatistics": [  
        {  
          "virtualHost": "iron.islands",  
          "virtualPort": "1234",  
          "protocol": "RFC",  
          "bytesReceived": 0,  
          "bytesSent": 0,  
          "calls": 0,  
          "resources": [  
            {  
              "resourceName": "TEST_FCT_1",  
              "enabled": true,  
              "bytesReceived": 0,  
              "bytesSent": 0,  
              "calls": 0  
            },  
            {  
              "resourceName": "TEST_FCT_2",  
              "enabled": true,  
              "bytesReceived": 0,  
              "bytesSent": 0,  
              "calls": 0  
            }  
          ]  
        },  
        {  
          "virtualHost": "the.north",  
          "virtualPort": "8080",  
          "protocol": "HTTP",  
          "bytesReceived": 1482,  
          "bytesSent": 705,  
          "calls": 3,  
          "mostRecentAccess": 1596810970078,  
          "resources": [  
            {  
              "resourceName": "/invoke/giants",  
              "enabled": true,  
              "bytesReceived": 0,  
              "bytesSent": 0,  
              "calls": 0  
            },  
            {  
              "resourceName": "/invoke/whiteWalkers",  
              "enabled": true,  
              "bytesReceived": 0,  
              "bytesSent": 0,  
              "calls": 0  
            },  
            {  
              "resourceName": "/invoke/nightWatch",  
              "enabled": true,  
              "bytesReceived": 1482,  
              "bytesSent": 705,  
              "calls": 3,  
              "mostRecentAccess": 1596810970078  
            }  
          ]  
        }  
      ]  
    }  
  ]  
}
```


Back to [Available APIs \[page 557\]](#)

Back to [Context \[page 556\]](#)

Master Role Check (available as of version 2.15.0)

With the master role check API, you can recognize if a Cloud Connector instance has currently the master role. The purpose of this master role check is only to recognize if the Cloud Connector instance is currently the master instance or not, without the need of providing credentials. It is a quick check that you can execute frequently.

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

Back to [Available APIs \[page 557\]](#)

Back to [Context \[page 556\]](#)

1.5.3.8 Alerting

Configure the Cloud Connector to send e-mail messages when situations occur that may prevent it from operating correctly.

To configure alert e-mails, choose [Alerting](#) from the top-left navigation menu.

You must specify the receivers of the alert e-mails ([E-mail Configuration](#)) as well as the Cloud Connector resources and components that you want to monitor ([Observation Configuration](#)). The corresponding [Alert Messages](#) are also shown in the Cloud Connector administration UI.

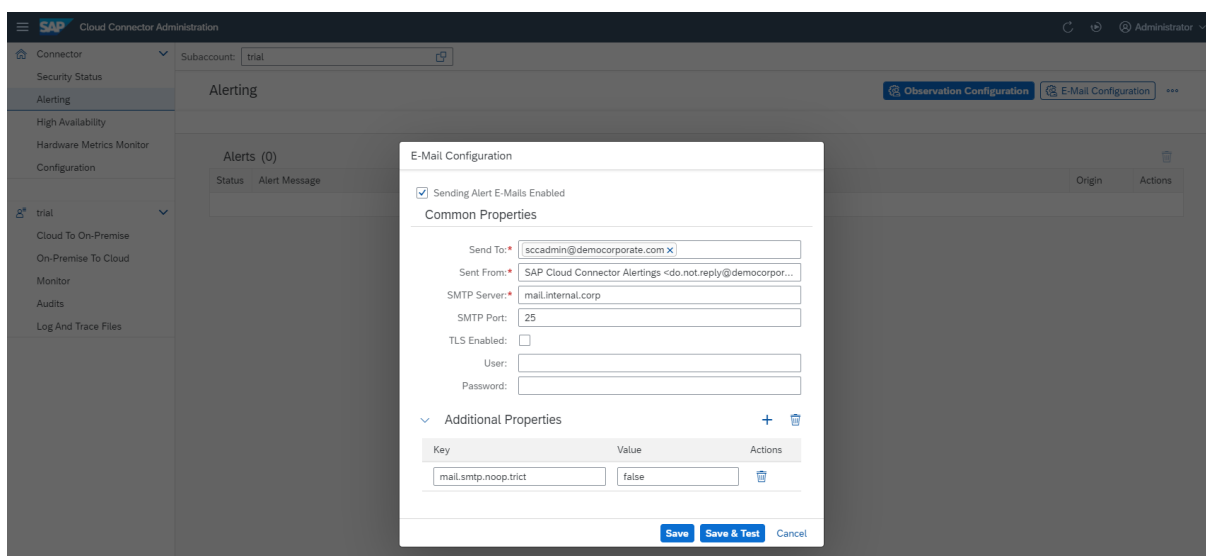
E-mail Configuration

1. Select [E-mail Configuration](#) to specify the list of em-ail addresses to which alerts should be sent ([Send To](#)).

ⓘ Note

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

2. Enter the sender's e-mail address (<Sent From>).
3. In <SMTP Server> provide the host of the mail server.
4. You can specify an <SMTP port>, if the server is not using the default ports. For details, contact your e-mail administrator or provider.
5. Mark the *TLS Enabled* check box if you want to establish a TLS-encrypted connection.
6. If the SMTP server requires authentication, provide <User> and <Password>.
7. In the *Additional Properties* section you can provide any property supported by the [Java Mail library](#) . All specified properties will be passed to the SMTP client.
8. Select *Save* to change the current configuration.



Note

Connections to an SMTP server over TLS can cause TLS errors if the SMTP server uses an "untrusted" certificate. If you cannot use a trusted certificate, you must import the public part of the issuer certificate to the JDK's trust storage.

Usually, the trust storage is done in the file *cacerts* in the Java directory (*jre/lib/security/cacerts*). For import, you can use the *keytool* utility:

```
keytool -import -storepass changeit -file <certificate used by SMTP server>
-keystore cacerts -alias <for example, SMTP_xyz>
```

For more information, see <https://docs.oracle.com/cd/E19830-01/819-4712/ablqw/index.html> .

Observation Configuration

Once you've entered the e-mail addresses to receive alerts, the next step is to identify the resources and components of the Cloud Connector: E-mail messages are sent when any of the chosen components or resources have malfunctioned or are in a critical state.

Note

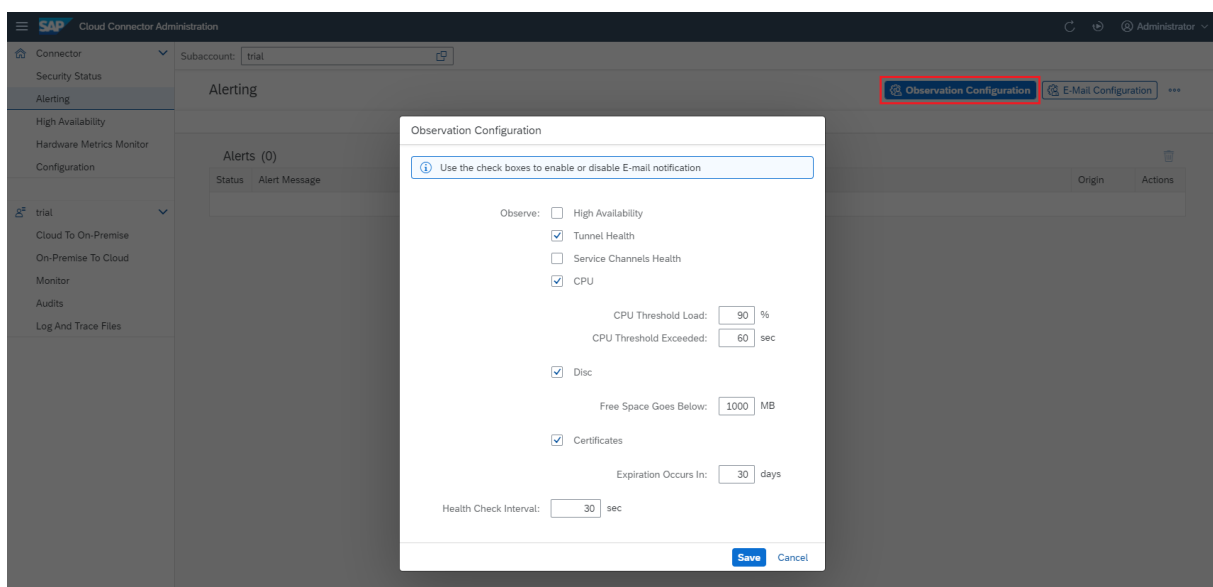
The Cloud Connector does not dispatch the same alert repeatedly. As soon as an issue has been resolved, an informational alert is generated, sent, and listed in [Alert Messages](#) (see section below).

1. Select [Observation Configuration](#) from the top-right corner of the window.
2. Select the components or resources you want to monitor.
 - [High Availability](#) alerts can occur in the context of an active high availability setup, meaning a shadow system is connected.
 - [Tunnel Health](#) and [Service Channels Health](#) refer to the state of the respective connections. Whenever such a connection is lost, an alert is triggered.

Note

These alerts are only triggered in case of an error or exception, but not upon intentional disconnect action.

- An excessively high [CPU](#) load over an extended period of time adversely affects performance and may be an indicator of serious issues that jeopardize the operability of the Cloud Connector. The CPU load is monitored and an alert is triggered whenever the CPU load exceeds and continues to exceed a given threshold percentage (the default is 90%) for more than a given period of time (the default is 60 seconds).
 - Although the Cloud Connector does not require nor consume large amounts of [Disk](#) space, running out of it is a circumstance that you should avoid. We recommend that you configure an alert to be sent if the disk space falls below a critical value (the default is 10 megabytes).
 - The Cloud Connector configuration contains various [Certificates](#). Whenever one of those expires, scenarios might no longer work as expected so it's important to get notified about the expiration (the default is 30 days).
3. (Optional) Change the [Health Check Interval](#) (the default is 30 seconds).
 4. Select [Save](#) to change the current configuration.



Alert Messages

The Cloud Connector shows alert messages also on screen, in [Alerting > Alert Messages](#).

You can remove alerts using [Delete](#) or [Delete All](#). If you delete active (unresolved) alerts, they reappear in the list after the next health check interval.

1.5.3.9 Audit Logging

Audit log data can alert Cloud Connector administrators to unusual or suspicious network and system behavior.

Additionally, the audit log data can provide auditors with information required to validate security policy enforcement and proper segregation of duties. IT staff can use the audit log data for root-cause analysis following a security incident.

The Cloud Connector includes an auditor tool for viewing and managing audit log information about access between the cloud and the Cloud Connector, as well as for tracking of configuration changes done in the Cloud Connector. The written audit log files are digitally signed by the Cloud Connector so that their integrity can be checked, see [Manage Audit Logs \[page 576\]](#).

Note

We recommend that you permanently switch on Cloud Connector audit logging in productive scenarios.

- Under normal circumstances, set the logging level to **Security** (the default configuration value).
- If legal requirements or company policies dictate it, set the logging level to **All**. This lets you use the log files to, for example, detect attacks of a malicious cloud application that tries to access on-premise services without permission, or in a forensic analysis of a security incident.

We also recommend that you regularly copy the audit log files of the Cloud Connector to an external persistent storage according to your local regulations. The audit log files can be found in the Cloud Connector root directory `/log/audit/<subaccount-name>/audit-log_<timestamp>.csv`.

1.5.3.9.1 Manage Audit Logs

Configure audit log settings and verify the integrity of audit logs.

Configure Audit Logs in the Cloud Connector

Choose [Audit](#) from your subaccount menu and go to [Settings](#) to specify the type of audit events the Cloud Connector should log at runtime. You can currently select between the following [Audit Levels](#) (for either `<subaccount>` and `<cross-subaccount>` scope):

- **Security:** Default value. The Cloud Connector writes an audit entry (`Access Denied`) for each request that was blocked. It also writes audit entries, whenever an administrator changes one of the critical configuration settings, such as exposed back-end systems, allowed resources, and so on.
- **All:** The Cloud Connector writes one audit entry for each received request, regardless of whether it was allowed to pass or not (`Access Allowed` and `Access Denied`). It also writes audit entries that are relevant to the **Security** mode.
- **Off:** No audit entries are written.

⚠ Caution

To prevent a single person from being able to both change the audit log level, and delete audit logs, we recommend that the operating system administrator and the SAP BTP administrator are different persons. We also suggest that you turn on the audit log at the operating system level for file operations.

→ Tip

We recommend that you don't log all events unless you are required to do so by legal requirements or company policies. Generally, logging security events only is sufficient.

To enable automatic cleanup of audit log files, choose a period (14 to 365 days) from the list in the field `<Automatic Cleanup>`.

Audit entries for configuration changes are written for the following categories:

- **Account:** A subaccount configuration was changed.
- **Configuration:** A new subaccount was added or a disaster recovery switch happened.
- **BackendMapping:** Changes to the virtual to internal system mappings.
- **AllowedResource:** In a virtual system, changes in the accessible resources.
- **DomainMapping:** Changes to the domain mappings.
- **ServiceChannelConfiguration:** The configuration of a service channel was changed.
- **SCCPassword:** The Cloud Connector administration password was changed.
- **SCCUser:** The Cloud Connector administration user was changed.
- **LDAPConfiguration:** Changes to the LDAP settings.
- **EmailConfiguration:** The Email settings for alerts were changed.
- **AlertConfiguration:** The observation settings for alerts were changed.
- **ScimConfiguration:** Something changed in the settings for the cloud user store.
- **KerberosConfiguration:** The Kerberos configuration was changed.
- **SNCSSettings:** 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 or download the selected audit entries.

- In the [Audit Type](#) field, you can select the types of auditing events you are interested in.
- In the [Pattern](#) field, you can specify a certain string that the detail text of each selected audit entry must contain. The detail text may contain, for example, information about the user name, requested resource/URL, or the virtual `<host>: <port>`. Basic wildcards (glob patterns) are supported, that is, asterisk for any number of characters (including none) and question mark for a single character. Use this feature to do the following:
 - Filter the audit log for all requests that a particular HTTP user has made during a certain time frame.
 - Identify all users who attempted to request a particular URL.
 - Identify all requests to a particular back-end system.
 - Determine whether a user has changed a certain Cloud Connector configuration. For example, a search for string `BackendMapping` returns all `add-`, `delete-` or `modify-` operations on the [Mapping Virtual To Internal System](#) page.
- The [Time Range](#) settings specify the time frame to which you want to limit the eligible audit entries.

These filter criteria are combined with a logical **AND** so that all audit entries that match these criteria are shown. If you have modified the criteria, choose the [Search](#) button again to display the updated selection of audit events that match the new criteria.

Use the [Download](#) button to download the selected audit entries as a compressed (GZIP) CSV file that can be imported, for example, by Excel. The ZIP file also contains a second text file, a manifest or information, that provides the selection parameters as well as the date and time when the selection was extracted.

Example

In the following example, the [Audit Viewer](#) displays `Any` audit entries, at `Security` level, for the time frame between **December 18 2020, 00:00:00** and **December 19, 00:00:00**:

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:///@hana.ondemand.com

Verify the Integrity of Audit Logs

To check the integrity of all or a part of the audit logs, go to `<scs_installation>/auditor`. This directory contains an executable `go` script file (respectively, `go.cmd` on Microsoft Windows and `go.sh` on other operating systems).

If you start the `go` file without specifying parameters from `<scs_installation>/auditor`, all available audit logs for the current Cloud Connector installation are verified.

The auditor tool is a Java application, and therefore requires a Java runtime, specified in `JAVA_HOME`, to execute:

- For Microsoft Windows OS, set `JAVA_HOME=<path-to-java-installation>`
- For Linux OS and Mac OS X, export: `JAVA_HOME=<path-to-java-installation>`

Alternatively, to execute Java, you can include the Java `bin` directory in the `PATH` variable.

You can check audit logs also in the UI, using the rightmost button of the **Audits** section.

Note

The selected date range determines the audit logs that are going to be checked (entire days only, ignoring the time of day).

Change the Location of Audit Logs

As of Cloud Connector 2.14 you can move audit logs to a different location. Standard location remains `log/audit`.

Note

Make sure there is enough space left on the device for the desired location and the Cloud Connector OS user has permission to write files to that location.

If you want to do this, proceed as follows:

1. Shut down the Cloud Connector.
2. Execute the respective script for the location change.
 1. For Microsoft Windows OS: `changeAuditLogPath.bat <desiredLocation>`.
 2. For Linux OS and Mac OS X: `./changeAuditLogPath.sh <desiredLocation>`.
3. The script checks if the target might be a network location. If this is assumed, the script asks for confirmation. Afterwards, it tries to move all existing audit logs to the new location. Only after successful move, the location change takes effect, otherwise it remains in the old place.
4. Start the Cloud Connector again.

Caution

If you choose a network location while access to the file system is slow, overall processing performance of the Cloud Connector may decrease significantly.

1.5.3.10 Troubleshooting

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

Note

For information about a specific problem or an error you have encountered, see [Connectivity Support \[page 615\]](#).

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. The left sidebar contains a navigation menu with 'Connector' selected. The main area displays 'Tunnel Information' for a subaccount named 'trial'. It shows the status as 'Connected', Tunnel ID as 'account:///', and Remote Name as 'connectivitynotification.hana.ondemand.com'. Below this, a table titled 'Cloud Connections' is highlighted with a red box. The table has columns: Application, Connections, Connected Since, Peer Labels, and Actions. It contains one entry for 'jcodemo' with 1 connection, connected since '18. Dezember 2020 14:41:46 MEZ', and peer labels '.....'.

The provided information includes:

- **Application name:** The name of the application, as also shown in the cockpit, for your subaccount
- **Connections:** The number of currently existing connections to the application
- **Connected Since:** The earliest start time of a connection to this application
- **Peer Labels:** The name of the application processes, as also shown for this application in the cockpit, for your subaccount

Log and Trace Settings

The [Log and Trace Files](#) page includes some files for troubleshooting that are intended primarily for SAP Support. These files include information about both internal Cloud Connector operations and details about the communication between the local and the remote (SAP BTP) tunnel endpoint.

If you encounter problems that seem to be caused by some trouble in the communication between your cloud application and the on-premise system, choose [Log and Trace Files](#) from your [Subaccount](#) menu, go to section [Settings](#), and activate the respective traces by selecting the [Edit](#) button:

- **Cloud Connector Loggers** adjusts the levels for Java loggers directly related to Cloud Connector functionality.
- **Other Loggers** adjusts the log level for all other Java loggers available at the runtime. Change this level only when requested to do so by SAP support. When set to a level higher than **Information**, it generates a large number of trace entries.
- **CPIC Trace Level** allows you to set the level between 0 and 3 and provides traces for the CPIC-based RFC communication with ABAP systems.
- When the **Payload Trace** is activated for a subaccount, all the HTTP and RFC traffic crossing the tunnel for that subaccount going through this Cloud Connector, is traced in files with names `traffic_trace_<subaccount id>_on_<regionhost>.trc`. This is helpful if you need to understand what documents have been exchanged between the involved systems.
- **Payload SNC Trace:** When the **Payload SNC trace** is activated for an account, all RFC SNC-based traffic crossing a service channel for that account (going through this Cloud Connector), is traced in files with names `payload_snc_trace_<account id>_on_<landscapehost>.trc`. This is helpful if you need to understand issues with SNC termination in the Cloud Connector.

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

Save

Cancel

⚠ Caution

Use any **payload and CPIC tracing at level 3** carefully, and only when requested to do so for support reasons. These traces may write sensitive information (such as payload data of HTTP/RFC requests and responses) to the trace files, and thus present a potential security risk. The Cloud Connector supports the implementation of a "four-eyes principle" for activating the trace levels that dump the network traffic into a trace file. This principle requires two users to activate a trace level that records traffic data.

For more information, see [Secure the Activation of Traffic Traces \[page 545\]](#).

Change the Location of Trace Files

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

📘 Note

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

Note

Make sure there is enough space left on the device for the desired location and the Cloud Connector OS user has permission to write files to that location.

If you want to do this, proceed as follows:

1. Shut down the Cloud Connector.
2. Execute the respective script for the location change.
 1. For Microsoft Windows OS: `changeLogAndTracePath.bat <desiredLocation>`.
 2. For Linux OS and Mac OS X: `./changeLogAndTracePath.sh <desiredLocation>`.
3. The script checks if the target might be a network location. If this is assumed, the script asks for confirmation. Afterwards, it tries to move the existing current trace file to the new location. Only after successful move, the location change takes effect. Otherwise, the file remains in the old place.
4. Start the Cloud Connector again.

Caution

If you choose a network location while access to the file system is slow, overall processing performance of the Cloud Connector may decrease significantly.

Log and Trace Files

View all existing trace files and delete the ones that are no longer needed.

The screenshot shows the SAP Cloud Connector Administration web interface. The left sidebar contains a navigation menu with options: Connector, Security Status, Alerting, High Availability, Hardware Metrics Monitor, Configuration, trial (selected), Cloud To On-Premise, On-Premise To Cloud, Monitor, Audits, and Log And Trace Files (highlighted). The main content area is titled 'Log And Trace Files' and includes buttons for 'Thread Dump', 'Guided Answers', 'Support Log Assistant', and a help icon. Below this is a 'Settings' section with a list of loggers and their configurations: Cloud Connector Loggers (Information), Other Loggers (Information), CPIC Trace Level (0), Payload Trace (No), SSL Trace (No), and Automatic Cleanup (Never). At the bottom, there is a table titled 'Log And Trace Files (4)' with columns for 'Last Modified', 'File Name', 'File Size', and 'Actions'. The table lists four log 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 icons for download, share, and delete. Below the table is a summary bar for 'LJS Log Summary' with buttons for 'All', '10', '12', and '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	Download Share Delete
2020-12-18 14:41:46 +0100	ljs_trace.log	59.8 KB	Download Share Delete
2020-12-18 14:11:00 +0100	vm_31400_gc.prf	0 Bytes	Download Share Delete
2020-12-17 15:51:04 +0100	localhost_http_access_2020-12-17.log	23.7 KB	Download Share Delete

LJS Log Summary: [All](#) [10](#) [12](#) [175](#)

To prevent your browser from being overloaded when multiple large files are loaded simultaneously, the Cloud Connector loads only one page into memory. Use the page buttons to move through the pages.

Use the [Download](#)/[Download All](#) icons to create a ZIP archive containing one trace file or all trace files. Download it to your local file system for convenient analysis.

Note

If you want to download more than one file, but not all, select the respective rows of the table and choose [Download All](#).

When running the Cloud Connector with SAP JVM or as of version 2.14 also with other JVMs, you can trigger the creation of a thread dump by choosing the [Thread Dump](#) button, which will be written to the JVM trace file `log/vm_$PID_trace.log` for SAP JVM and `log/vm_$PID_threads.log` for other JVMs. You may be asked by SAP support to create one, if considered helpful during incident analysis.

Note

From the UI, you can't delete trace files that are currently in use. You can delete them from the Linux OS command line; however, we recommend that you do not use this option to avoid inconsistencies in the internal trace management of the Cloud Connector.

Two buttons may be helpful to solve issues on your own:

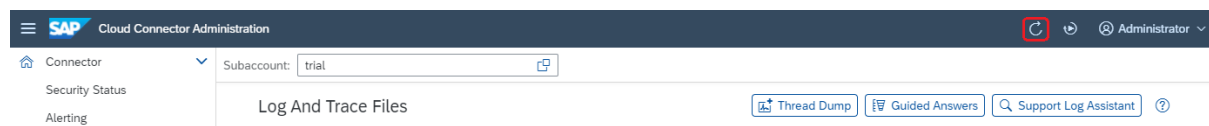
- [Guided Answers](#): A new tab or window opens, showing the Cloud Connector section in [Guided Answers](#). It helps you identify many issues that are classified through hierarchical topics. Once you found a matching issue, a solution is provided either directly, or by references to SAP Help Portal, Knowledge Base Articles (KBAs), and SAP notes.
- [Support Log Assistant](#): Opens the support log assistant. There, you can upload Cloud Connector log files and have them analyzed. After triggering the scan, the tool lists all issues for which a solution can be identified.

Note

The support log assistant analyzes the complete log. Therefore, also older issues may be found that are no longer relevant.

Once a problem has been identified, you should turn off the trace again by editing the trace and log settings accordingly to not flood the files with unnecessary entries.

Use the [Refresh](#) button to update the information that appears. For example, you can use this button because more trace files might have been written since you last updated the display.



Error Analysis and Support: Which Logs are Relevant?

If you contact SAP support for help, please always attach the appropriate log files and provide the timestamp or period, when the reported issue was observed. Depending on the situation, different logs may help to find the root cause.

Some typical settings to get the required data are listed below:

- **<Cloud Connector Loggers>** provide **details related to connections to SAP BTP and to backend systems as well as master-shadow communication in case of a high availability setup**. However, it does not contain any payload data. This kind of trace is written into `ljs_trace.log`, which is the most relevant log for the Cloud Connector.
- **<Other Loggers>** provide **details related to the tomcat runtime**, in which the Cloud Connector is running. The traces are written into `ljs_trace.log` as well, but they are needed only in very special support situations. If you don't need these traces, leave the level on `Information` or even lower.
- **Payload data** are written into the traffic trace file for HTTP or RFC requests if the payload trace is activated, or into the CPI-C trace file for RFC requests, if the CPI-C trace is set to level 3.
- **Payload SNC data** are written into a payload SNC trace file for incoming SNC RFC requests if SNC payload trace is activated.
- **<TLS trace>** is helpful to **analyze TLS handshake failures** from Cloud Connector to Cloud or from Cloud Connector to backend. It should be turned off again as soon as the issue has been reproduced and recorded in the traces.
- Setting the audit log on level `ALL` for **<Subaccount Audit Level>** is the easiest way to **check if a request reached the the Cloud Connector and if it is being processed**.

Related Information

[Getting Support](#)

1.5.3.11 Process Guidelines for Hybrid Scenarios

A hybrid scenario is one, in which applications running on SAP BTP require access to on-premise systems. Define and document your scenario to get an overview of the required process steps.

Tasks

[Document the Landscape of a Hybrid Solution \[page 586\]](#)

[Document Administrator Roles \[page 586\]](#)

[Document Communication Channels \[page 587\]](#)

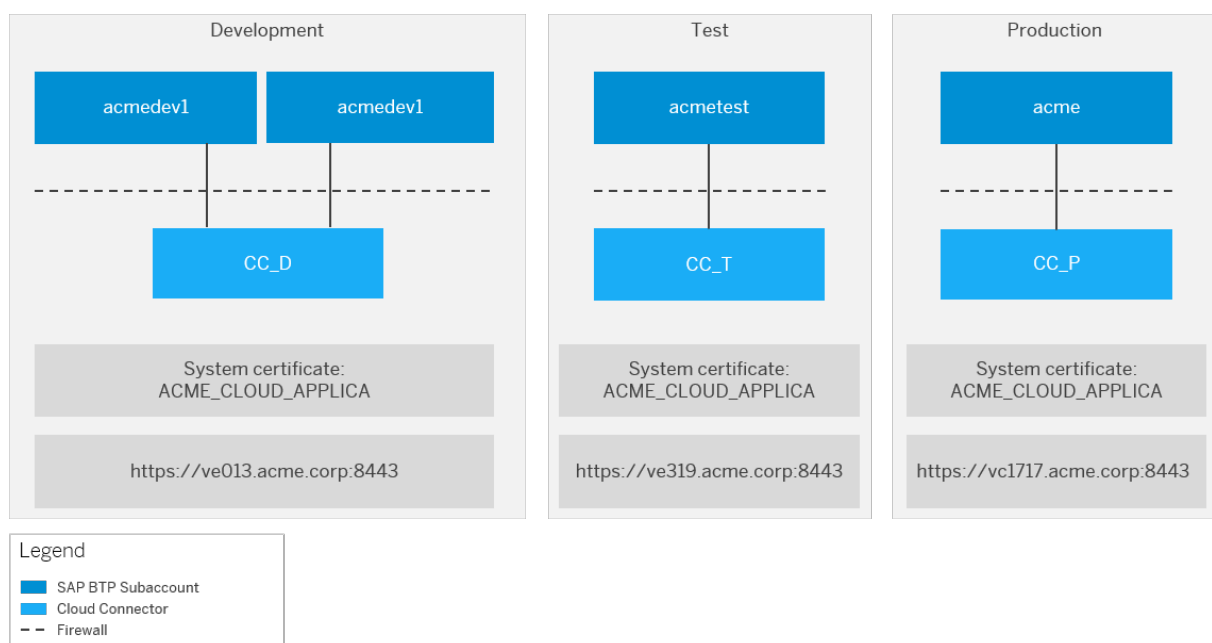
[Define Project and Development Guidelines \[page 587\]](#)

[Define How to Set a Cloud Application Live \[page 588\]](#)

Document the Landscape of a Hybrid Solution

To gain an overview of the cloud and on-premise landscape that is relevant for your hybrid scenario, we recommend that you diagrammatically document your cloud subaccounts, their connected Cloud Connectors and any on-premise back-end systems. Include the subaccount names, the purpose of the subaccounts (dev, test, prod), information about the Cloud Connector machines (host, domains), the URLs of the Cloud Connectors in the landscape overview document, and any other details you might find useful to include.

An example of landscape overview documentation could look like this:



[Back to Tasks \[page 585\]](#)

Document Administrator Roles

Document the users who have administrator access to the cloud subaccounts, to the Cloud Connector operating system, and to the Cloud Connector administration UI.

Such an administrator role documentation could look like following sample table:

Resource	bernardo@acme.com	mary@acme.com	smitha@acme.com	greg@acme.com
Cloud Subaccount (CA) Dev1	X			
CA Dev2		X		

Resource	bernardo@acme.com	mary@acme.com	smitha@acme.com	greg@acme.com
CA Test			X	X
CA Prod				X
Cloud Connector Dev1 + Dev2	X	X		
Cloud Connector Test			X	X
Cloud Connector Prod				X
Cloud Connector Dev1 + Dev2 file system			X	X
Cloud Connector Test file system				X
Cloud Connector Prod file system				

Back to [Tasks \[page 585\]](#)

Document Communication Channels

Create and document separate email distribution lists for both the cloud subaccount administrators and the Cloud Connector administrators.

An example of the documented communication channels could look like this:

Landscape	Distribution List
Cloud Subaccount Administrators	DL ACME BTP Subaccount Admins
Cloud Connector Administrators	DL ACME Cloud Connector Admins

Back to [Tasks \[page 585\]](#)

Define Project and Development Guidelines

Define and document mandatory project and development guidelines for your SAP BTP projects. An example of such a guideline could be similar to the following.

Every SAP BTP project in this organization requires the following:

- Use Maven, Nexus, Git-&-Gerrit for the application development.
- Align with accountable manager in projects (including the names).
- Align with accountable security officer in projects (including the names).

- For externally developed source code, an official handover to the organization.
- Fulfill connection restrictions in a three-system landscape, that is, use a staged landscape for dev, test and prod, and, for example, the dev landscape connects only to dev systems, and so on.
- Productive subaccounts cannot use the same Cloud Connector as a dev or test subaccount.

Back to [Tasks \[page 585\]](#)

Define How to Set a Cloud Application Live

Define and document how to set a cloud application live and how to configure needed connectivity for such an application.

For example, the following processes could be seen as relevant and should be defined and document in more detail:

1. Transferring application to production: Steps for transferring an application to the productive status on the SAP BTP.
2. Application connectivity: The steps for adding a connectivity destination to a deployed application for connections to other resources in the test or productive landscape.
3. Cloud Connector Connectivity: Steps for adding an on-premise resource to the Cloud Connector in the test or productive landscapes to make it available for the connected cloud subaccounts.
4. On-premise system connectivity: The steps for setting up a trusted relationship between an on-premise system and the Cloud Connector, and to configure user authentication and authorization in the on-premise system in the test or productive landscapes.
5. Application authorization: The steps for requesting and assigning an authorization that is available inside the SAP BTP application to a user in the test or productive landscapes.
6. Administrator permissions: Steps for requesting and assigning the administrator permissions in a cloud subaccount to a user in the test or productive landscape.

Back to [Tasks \[page 585\]](#)

1.5.3.12 Configuring Backup

Find an overview of backup procedures for the Cloud Connector.

Configuration Backup [page 511]	Backup and restore your Cloud Connector configuration via the administration UI.
Backup [page 412]	Manage the Cloud Connector's configuration backup via REST API.
Backup And Restore Configuration [page 486]	Example: Backup and restore the Cloud Connector configuration via REST API.

1.5.4 Security

Learn how Cloud Connector features help you manage security.

Features

Security is a crucial concern for any cloud-based solution. It has a major impact on the business decision of enterprises whether to make use of such solutions. SAP BTP is a platform-as-a-service offering designed to run business-critical applications and processes for enterprises, with security considered on all levels of the on-demand platform:

Level	Features
Application Layer [page 590]	<ul style="list-style-type: none">• Frontend security• Security standard-based application development
Service Layer [page 590]	<ul style="list-style-type: none">• Identity and access management• Data protection• Regulatory compliance management
Cloud Infrastructure Layer [page 595]	<ul style="list-style-type: none">• Network infrastructure and communication• Sandboxing• Intrusion detection and prevention
Physical and Environmental Layer [page 596]	<ul style="list-style-type: none">• Strict physical access control• High availability

The Cloud Connector enables integration of cloud applications with services and systems running in customer networks, and supports database connections from the customer network to SAP HANA databases running on SAP BTP. As these are security-sensitive topics, this section gives an overview on how the Cloud Connector helps maintain security standards for the mentioned scenarios.

Target Audience

The content of this section concerns:

- System and IT administrators
- Technology consultants
- Solution architects

Related Information

[Security Guidelines \[page 596\]](#)

1.5.4.1 Application Layer

On application level, the main tasks to ensure secure Cloud Connector operations are to provide appropriate frontend security (for example, validation of entries) and a secure application development.

Product Security Standard

Basically, you should follow the rules given in the product security standard, for example, protection against cross-site scripting (XSS) and cross-site request forgery (XSRF).

Scope and Design

The scope and design of security measures on application level strongly depend on the specific needs of your application.

1.5.4.2 Service Layer

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

Overview

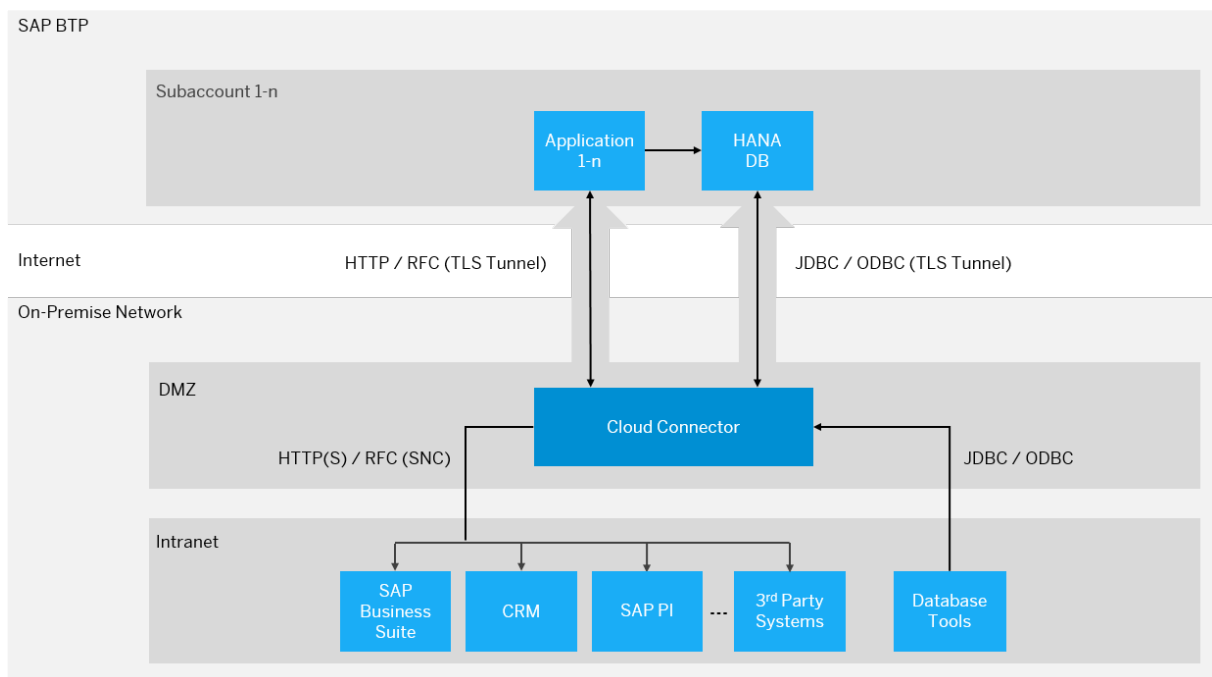
After installing the Cloud Connector as integration agent in your on-premise network, you can use it to establish a persistent TLS tunnel to SAP BTP subaccounts.

To establish this tunnel, the Cloud Connector administrator must authenticate himself or herself against the related SAP BTP subaccount of which he or she must be a member. Once established, the tunnel can be used by applications of the connected subaccount to remotely call systems in your network.

Architecture

The figure below shows a system landscape in which the Cloud Connector is used for secure connectivity between SAP BTP applications and on-premise systems.

- A single Cloud Connector instance can connect to multiple SAP BTP subaccounts, each connection requiring separate authentication and defining an own set of configuration.
- You can connect an arbitrary number of SAP and non-SAP systems to a single Cloud Connector instance.
- The on-premise system does not need to be touched when used with the Cloud Connector, unless you configure trust between the Cloud Connector and your on-premise system. A trust configuration is required, for example, for principal propagation (single sign-on), see [Configuring Principal Propagation \[page 304\]](#).
- You can operate the Cloud Connector in a high availability mode. To achieve this, you must install a second (redundant) Cloud Connector (shadow instance), which takes over from the master instance in case of a downtime.
- The Cloud Connector also supports the communication direction from the on-premise network to the SAP BTP subaccount, using a database tunnel that lets you connect common ODBC/JDBC database tools to SAP HANA as well as other available databases in SAP BTP.



Related Information

[Network Zones \[page 592\]](#)

[Inbound Connectivity \[page 592\]](#)

[Outbound Connectivity \[page 594\]](#)

1.5.4.2.1 Network Zones

Choosing a network zone for the Cloud Connector installation.

A company network is usually divided into multiple network zones according to the security level of the contained systems. The DMZ network zone contains and exposes the external-facing services of an organization to an untrusted network, typically the Internet. Besides this, there can be one or multiple other network zones which contain the components and services provided in the company's intranet.

You can set up the Cloud Connector either in the DMZ or in an inner network zone. Technical prerequisites for the Cloud Connector to work properly are:

- The Cloud Connector must have access to the SAP BTP landscape host, either directly or via HTTPS proxy (see also: [Prerequisites \[page 239\]](#)).
- The Cloud Connector must have direct access to the internal systems it shall provide access to. I.e. there must be transparent connectivity between the Cloud Connector and the internal system.

It's a company's decision, whether the Cloud Connector is set up in the DMZ and operated centrally by an IT department or set up in the intranet and operated by the line of business.

Related Information

[Network Zones \[page 256\]](#)

1.5.4.2.2 Inbound Connectivity

For inbound connections into the on-premise network, the Cloud Connector acts as a reverse invoke proxy between SAP BTP and the internal systems.

Exposing Resources

Once installed, none of the internal systems are accessible by default through the Cloud Connector: you must configure explicitly each system and each service and resource on every system to be exposed to SAP BTP in the Cloud Connector.

You can also specify a virtual host name and port for a configured on-premise system, which is then used in the cloud. Doing this, you can avoid that information on physical hosts is exposed to the cloud.

TLS Tunnel

The TLS (Transport Layer Security) tunnel is established from the Cloud Connector to SAP BTP via a so-called **reverse invoke** approach. This lets an administrator have full control of the tunnel, since it can't be established from the cloud or from somewhere else outside the company network. The Cloud Connector administrator is the one who decides when the tunnel is established or closed.

The tunnel itself is using TLS with strong encryption of the communication, and mutual authentication of both communication sides, the client side (Cloud Connector) and the server side (SAP BTP).

The X.509 certificates which are used to authenticate the Cloud Connector and the SAP BTP subaccount are issued and controlled by SAP BTP. They are kept in secure storages in the Cloud Connector and in the cloud. Having encrypted and authenticated the tunnel, confidentiality and authenticity of the communication between the SAP BTP applications and the Cloud Connector is guaranteed.

Restricting Allowed Applications

As an additional level of control, the Cloud Connector optionally allows restricting the list of SAP BTP applications which are able to use the tunnel. This is useful in situations where multiple applications are deployed in a single SAP BTP subaccount while only particular applications require connectivity to on-premise systems.

Isolation on Subaccount Level

SAP BTP guarantees strict isolation on subaccount level provided by its infrastructure and platform layer. An application of one subaccount is not able to access and use resources of another subaccount.

Supported Protocols

The Cloud Connector supports inbound connectivity for HTTP and RFC, any other protocol is not supported.

- The payload sent via these protocols is encrypted on TLS/tunnel-level.
- For the route from the Cloud Connector to the on-premise systems, Cloud Connector administrators have the choice for each configured on-premise system whether to use HTTP, HTTPS, RFC or RFC over SNC.
- For HTTPS, you can configure a so-called system certificate in the Cloud Connector which is used for the trust relationship between the Cloud Connector and the connected on-premise systems.
- For RFC over SNC, you can configure an SNC PSE in the Cloud Connector respectively.

Principal Propagation

The Cloud Connector also supports principal propagation of the cloud user identity to connected on-premise systems (single sign-on). For this, the system certificate (in case of HTTPS) or the SNC PSE (in case of RFC)

is mandatory to be configured and trust with the respective on-premise system must be established. Trust configuration, in particular for principal propagation, is the only reason to configure and touch an on-premise system when using it with the Cloud Connector.

Related Information

[Configuring Principal Propagation \[page 304\]](#)

1.5.4.2.3 Outbound Connectivity

The Cloud Connector supports the communication direction from the on-premise network to SAP BTP, using a database tunnel.

The database tunnel is used to connect local database tools via JDBC or ODBC to the SAP HANA DB or other databases on SAP BTP, for example, SAP Business Objects tools like Lumira, BOE or Data Services.

- The database tunnel only allows JDBC and ODBC connections from the Cloud Connector into the cloud. A reuse for other protocols is not possible.
- The tunnel uses the same security mechanisms as for the inbound connectivity:
 - TLS-encryption and mutual authentication
 - Audit logging

To use the database tunnel, two different SAP BTP users are required:

- A platform user (member of the SAP BTP subaccount) establishes the database tunnel to the HANA DB.
- A HANA DB user is needed for the ODBC/JDBC connection to the database itself. For the HANA DB user, the role and privilege management of HANA can be used to control which actions he or she can perform on the database.

Related Information

[Using Service Channels \[page 491\]](#)

1.5.4.2.4 Audit Log

As audit logging is a critical element of an organization's risk management strategy, the Cloud Connector provides audit logging for the complete record of access between cloud and Cloud Connector as well as of configuration changes done in the Cloud Connector.

Integrity Check

The written audit log files are digitally signed by the Cloud Connector so that they can be checked for integrity (see also: [Manage Audit Logs \[page 576\]](#)).

Alerting

The audit log data of the Cloud Connector can be used to alert Cloud Connector administrators regarding unusual or suspicious network and system behavior.

Additional Use Cases

- The audit log data can provide auditors with information required to validate security policy enforcement and proper segregation of duties.
- IT staff can use the audit log data for root-cause analysis following a security incident.

Related Information

[Audit Logging \[page 576\]](#)

1.5.4.3 Cloud Infrastructure Layer

Infrastructure and network facilities of the SAP BTP ensure security on network layer by limiting access to authorized persons and specific business purposes.

Isolated Network

The SAP BTP landscape runs in an isolated network, which is protected from the outside by firewalls, DMZ, and communication proxies for all inbound and outbound communications to and from the network.

Sandboxed Environments

The SAP BTP infrastructure layer also ensures that platform services, like the SAP BTP Connectivity, and applications are running isolated, in sandboxed environments. An interaction between them is only possible over a secure remote communication channel.

1.5.4.4 Physical and Environmental Layer

Learn about data center security provided for SAP BTP Connectivity.

SAP BTP runs in SAP-hosted data centers which are compliant with regulatory requirements. The security measures include, for example:

- strict physical access control mechanisms using biometrics, video surveillance, and sensors
- high availability and disaster recoverability with redundant power supply and own power generation

1.5.4.5 Security Guidelines

Find a checklist of recommended security measures for the Cloud Connector.

Topics

Hover over the elements for a description. Click an element to find the recommended actions in the table below.


Network Zone	Administration UI	High Availability
On-Premise Configuration	OS-Level Protection	Protocol Security
Audit Logging	Instances	

- [#unique_179/unique_179_Connect_42_network](#) [page 597]
- [#unique_179/unique_179_Connect_42_ui](#) [page 597]
- [#unique_179/unique_179_Connect_42_availability](#) [page 599]

- [#unique_179/unique_179_Connect_42_protocols \[page 599\]](#)
- [#unique_179/unique_179_Connect_42_os \[page 597\]](#)
- [#unique_179/unique_179_Connect_42_oP \[page 599\]](#)
- [#unique_179/unique_179_Connect_42_instances \[page 600\]](#)
- [#unique_179/unique_179_Connect_42_audit \[page 598\]](#)

Recommended Actions

Topic	Description	Recommended Action
Network Zone Back to Topics [page 596]	Depending on the needs of the project, the Cloud Connector can be either set up in the DMZ and operated centrally by the IT department or set up in the intranet and operated by the line-of-business.	To access highly secure on-premise systems, operate the Cloud Connector centrally by the IT department and install it in the DMZ of the company network. Set up trust between the on-premise system and the Cloud Connector, and only accept requests from trusted Cloud Connectors in the system.
OS-Level Protection Back to Topics [page 596]	The Cloud Connector is a security-critical component that handles the inbound access from SAP BTP applications to systems of an on-premise network. Methods to secure the operating system , on which the Cloud Connector is running, should be applied.	Restrict access to the operating system on which the Cloud Connector is installed to the minimal set of users who should administrate the Cloud Connector. Use the machine which runs the Cloud Connector only for this purpose and don't reuse it for other scenarios. Use hard-drive encryption for the machine that runs the Cloud Connector. This ensures that the Cloud Connector configuration data cannot be read or modified by unauthorized users, even if they obtain access to the hard drive. Turn on the audit log on operating system level to monitor the file operations.
Administration UI Back to Topics [page 596]	After installation, the Cloud Connector provides an initial user name and password and forces the user (Administrator) to change the password upon initial login.	Change the password of the Administrator user immediately after installation. Choose a strong password for the user (see also Recommendations for Secure Setup [page 272]).

Topic	Description	Recommended Action
		Configure a corporate LDAP system for the user management of the Cloud Connector administrator users. This guarantees that users of the Cloud Connector administration UI are named users and can be traced via the Cloud Connector audit log (see Use LDAP for User Administration [page 521]).
	<p>You can access the Cloud Connector administration UI remotely via HTTPS.</p> <p>After installation, it uses a self-signed X.509 certificate as TLS server certificate, which is not trusted by default by Web browsers.</p>	Exchange the self-signed X.509 certificate of the Cloud Connector administration UI by a certificate that is trusted by your company and the company's approved Web browser settings (see Exchange UI Certificates in the Administration UI [page 517]).
		For high-security scenarios, limit the access to the Cloud Connector administration UI to localhost (see also Recommendations for Secure Setup [page 272]).
		Use a JVM that allows to limit the ciphers to a set considered safe (see Recommendations for Secure Setup [page 272]). A list of cipher suites, which are currently considered safe, is available in Use Secure Cipher Suites  .
Audit Logging Back to Topics [page 596]	For end-to-end traceability of configuration changes in the Cloud Connector, as well as communication delivered by the Cloud Connector, switch on audit logging for productive scenarios.	<p>Switch on audit logging in the Cloud Connector: set audit level to "All" (see Recommendations for Secure Setup [page 272] and Manage Audit Logs [page 576])</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>

Topic	Description	Recommended Action
High Availability Back to Topics [page 596]	<p>To guarantee high availability of the connectivity for cloud integration scenarios, run productive instances of the Cloud Connector in high availability mode, that is, with a second (redundant) Cloud Connector in place.</p>	<p>Use the high availability feature of the Cloud Connector for productive scenarios (see Install a Failover Instance for High Availability [page 538]).</p>
Supported Protocols Back to Topics [page 596]	<p>HTTP, HTTPS, RFC and RFC over SNC are currently supported as protocols for the communication direction from the cloud to on-premise.</p> <p>The route from the application VM in the cloud to the Cloud Connector is always encrypted.</p> <p>You can configure the route from the Cloud Connector to the on-premise system to be encrypted or unencrypted.</p>	<p>The route from the Cloud Connector to the on-premise system should be encrypted using TLS (for HTTPS) or SNC (for RFC).</p> <p>Trust between the Cloud Connector and the connected on-premise systems should be established (see Set Up Trust [page 304]).</p>
Configuration of On-Premise Systems Back to Topics [page 596]	<p>When configuring the access to an internal system in the Cloud Connector, map physical host names to virtual host names to prevent exposure of information on physical systems to the cloud.</p>	<p>Use hostname mapping of exposed on-premise systems in the access control of the Cloud Connector (see Configure Access Control (HTTP) [page 342] and Configure Access Control (RFC) [page 350]).</p>
	<p>When configuring the access to an internal system, restrict access to those resources which are actually required by the cloud applications. Do not expose the complete system.</p>	<p>Narrow access to on-premise systems to resources required by the relevant cloud applications in the access control of the Cloud Connector (see Configure Access Control (HTTP) [page 342] and Configure Access Control (RFC) [page 350]).</p>
	<p>To allow access only for trusted applications of your SAP BTP subaccount to on-premise systems, configure the list of trusted applications in the Cloud Connector.</p>	<p>Narrow the list of cloud applications which are allowed to use the on-premise tunnel to the ones that need on-premise connectivity (see Set Up Trust [page 304]).</p>

Topic	Description	Recommended Action
Cloud Connector Instances Back to Topics [page 596]	<p>You can connect a single Cloud Connector instance to multiple SAP BTP subaccounts.</p> <p>Subaccounts can be created by SAP BTP users as a self service. Different subaccounts are often used to separate development, test and production.</p> <p>Do not mix productive Cloud Connector usages with development or test scenarios.</p>	Use different Cloud Connector instances to separate productive and non-productive scenarios.

Related Information

[Recommendations for Secure Setup \[page 272\]](#)
[Secure the Activation of Traffic Traces \[page 545\]](#)

1.5.5 Upgrade

Upgrade your Cloud Connector and avoid connectivity downtime during the update.

The steps for upgrading your Cloud Connector are specific to the operating system that you use. Previous settings and configurations are automatically preserved.

⚠ Caution

Upgrade is supported only for *installer* versions, not for *portable* versions, see [Installation \[page 238\]](#). Before upgrading, please check the [Prerequisites \[page 239\]](#) and make sure your environment fits the new version. We recommend that you create a [Configuration Backup \[page 511\]](#) before starting an upgrade.

Avoid Connectivity Downtime

If you have a single-machine Cloud Connector installation, a short downtime is unavoidable during the upgrade process. However, if you have set up a master and a shadow instance, you can perform the upgrade without downtime by executing the following procedure:

1. Shut down the shadow instance.
2. Perform the upgrade on the shadow instance. (Follow the relevant procedure below.)

3. Restart the shadow instance and wait until it has connected to the master instance.
4. Perform a [Switch Roles](#) operation by pressing the corresponding button in the master administration UI. The master instance has now changed into a shadow instance.

⚠ Caution

After upgrading the former shadow instance from a version prior to 2.13 and having switched its role to be the new master instance, reset high availability settings in *both* instances *now*, before continuing to upgrade the second instance from a version prior to 2.13 as well. The master-shadow connection must be re-established after both instances have been upgraded from versions prior to 2.13 to versions 2.13 or higher.

5. Shut down the new shadow instance and perform the upgrade procedure on it as well.
6. Restart the new shadow instance and wait until it has connected to the already upgraded current master instance.
7. Perform again the [Switch Roles](#) operation if you want the previous master instance to act as the new master instance again.

Result: Both instances have now been upgraded without connectivity downtime and without configuration loss.

For more information, see [Install a Failover Instance for High Availability \[page 538\]](#).

Microsoft Windows OS

1. Uninstall the Cloud Connector as described in [Uninstallation \[page 603\]](#) and make sure to retain the existing configuration.
2. Reinstall the Cloud Connector within the same directory. For more information, see [Installation on Microsoft Windows OS \[page 265\]](#).
3. Before accessing the administration UI, clear your browser cache to avoid any unpredictable behavior due to the upgraded UI.

Linux OS

1. Execute the following command: :

```
rpm -U com.sap.scc-ui-<version>.rpm
```

📘 Note

Daemon extensions (as of Cloud Connector version 2.12.3)

All extensions to the daemon provided via `scc_daemon_extension.sh` mechanism will survive a version update. An upgrade to version 2.12.3 will already consider an existing file, even though previous versions were not supporting that feature.

2. Before accessing the administration UI, clear your browser cache to avoid any unpredictable behavior due to the upgraded UI.

1.5.6 Update the Java VM

How to update the Java VM used by the Cloud Connector.

Sometimes you must update the Java VM used by the Cloud Connector, for example, because of expired TLS certificates contained in the JVM trust store, bug fixes, deprecated JVM versions, and so on.

- If you make a replacement in the same directory, shut down the Cloud Connector, upgrade the JVM, and restart the Cloud Connector when you are done.
- **If you change the installation directory of the JVM, follow the steps below** for your operating system.

Make sure that the JVM has been installed successfully.

Note

A Java Runtime Environment (JRE) is not sufficient. You must use a JDK or SAP JVM.

Microsoft Windows OS

1. Make sure that the current user has administrative privileges.
2. Shutdown the Cloud Connector (for example, by stopping the corresponding Windows service or by double-clicking the [Stop SAP Cloud Connector](#) shortcut).
3. Open the *registry editor* (regedit32) and locate the following registry entry:
HKEY_LOCAL_MACHINE\SOFTWARE\SAP\Cloud Connector.
4. Change the value JavaHome to reflect the installation directory of the new SAP JVM or JDK.

Note

The *bin* subdirectory must not be part of the JavaHome value.

If the JavaHome value does not yet exist, create it here with a "String Value" (REG_SZ) and specify the full path of the Java installation directory, for example: C:\Program Files\sapjvm.

5. Close the registry editor and restart the Cloud Connector.

Linux OS

1. Open a shell command line prompt as root user.
2. In this shell, set the environment variable JAVA_HOME, pointing to the installation directory of the new SAP JVM or JDK, for example:
 - in csh/tcsh:

```
setenv JAVA_HOME /opt/sap/sapjvm_8
```
 - in sh/bash/dash/zsh:

```
export JAVA_HOME=/opt/sap/sapjvm_8
```

3. Execute the command

```
System V init distributions: service scc_daemon stop  
systemd distributions: systemctl stop scc_daemon
```

4. Execute the command

```
System V init distributions: /opt/sap/scc/daemon.sh reinstall  
systemd distributions: /opt/sap/scc/daemon.sh reinstallSystemd
```

5. Execute the command

```
System V init distributions: service scc_daemon start  
systemd distributions: systemctl start scc_daemon
```

Both Operating Systems

Note

- If you use your own CA certificates for the Email configuration (see [Alerting \[page 573\]](#)) or for LDAP (see [Use LDAP for User Administration \[page 521\]](#)), you must reimport them to the JVM trust store as described there.
- Make sure the selected cipher suites are accepted by the JVM you are about to install. If in doubt, revert to the default selection prior to changing the JVM.

After executing the above steps, the Cloud Connector should be running again and should have picked up the new Java version during startup. You can verify this by logging in to the Cloud Connector with your favorite browser, opening the [About](#) dialogue and checking that the field **<Java Details>** shows the version number and build date of the new Java VM. After you verified that the new JVM is indeed used by the Cloud Connector, delete or uninstall the old JVM.

1.5.7 Uninstallation

Uninstall an installer version or portable version of the Cloud Connector.

- If you have installed an installer variant of the Cloud Connector, follow the steps for your operating system to uninstall the Cloud Connector.
- To uninstall a developer version, proceed as described in section **Portable Variants**.

Microsoft Windows OS

1. In the Windows software administration tool, search for `Cloud Connector` (formerly named `SAP HANA cloud connector 2.x`).
2. Select the entry and follow the appropriate steps to uninstall it.
3. When you are uninstalling in the context of an upgrade, make sure to retain the configuration files.

Linux OS

To uninstall Cloud Connector 2.x, execute the following command:

```
rpm -e com.sap.scc-ui
```

⚠ Caution

This command also removes the configuration files.

Mac OS X

There is no installer variant for Mac OS X, only a `portable` one.

Portable Variants

(Microsoft Windows OS, Linux OS, Mac OS X) If you have installed a portable version (zip or tgz archive) of the Cloud Connector, simply remove the directory in which you have extracted the Cloud Connector archive.

Related Information

[Installation \[page 238\]](#)

1.5.8 Frequently Asked Questions

Answers to the most common questions about the Cloud Connector.

Technical Issues

Does the Cloud Connector send data from on-premise systems to SAP BTP or the other way around?

The connection is opened from the on-premise system to the cloud, but is then used in the other direction.

An on-premise system is, in contrast to a cloud system, normally located behind a restrictive firewall and its services aren't accessible thru the Internet. This concept follows a widely used pattern often referred to as *reverse invoke proxy*.

Is the connection between the SAP BTP and the Cloud Connector encrypted?

Yes, by default, TLS encryption is used for the tunnel between SAP BTP and the Cloud Connector.

If used properly, TLS is a highly secure protocol. It is the industry standard for encrypted communication and also, for example, as a secure channel in HTTPS.

Keep your Cloud Connector installation and JDK updated to avoid the use of weak and deprecated ciphers for TLS communication. Which cipher and TLS versions are actually used, is defined by both the cloud region setup and the JDK that is used for Cloud Connector. The TLS implementation used for the communication is the one of the JDK.

Can I use a TLS-terminating firewall between Cloud Connector and SAP BTP?

This is not possible. Basically, this is a desired man-in-the-middle attack, which does not allow the Cloud Connector to establish a mutual trust to the SAP BTP side.

Can I copy/clone a Cloud Connector installation and use it in parallel on a different machine?

This is not supported. You would face issues regularly, as those two instances will be considered as one, which is not expected by the cloud side.

If you just want to move the installation to a new machine, create a backup via the [Configuration Backup \[page 511\]](#) feature, create a new installation, import the backup, and discard the previous installation.

What is the oldest version of SAP Business Suite that's compatible with the Cloud Connector?

The Cloud Connector can connect an SAP Business Suite system version 4.6C and newer.

Which Java versions are supported to run the Cloud Connector?

Supported Java Version				
6	7	8	11	17

Cloud Connector Ver- sion	< 2.7.2	Yes	Yes	No	No	No
	= 2.7.2	Yes	Yes	Yes	No	No
	>= 2.8	No	Yes	Yes	No	No
	>= 2.12.3	No	No	Yes	No	No
	>= 2.14.0	No	No	Yes	Yes	No
	>= 2.15.0	No	No	Yes	Yes	Yes

⚠ Restriction

Support for Java 7 has been discontinued. For more information, see [Prerequisites \[page 241\]](#).

→ Tip

We recommend that you always use the latest patch level of the respective Java version.

⚠ Caution

Version 2.8 and later of the Cloud Connector may have problems with ciphers in Google Chrome, if you use the JVM 7. For more information read [this SCN Article](#).

Which configuration in the SAP BTP destinations do I need to handle the user management access to the Cloud User Store of the Cloud Connector?

See [Configure an On-Premise User Store \[page 489\]](#).

Is the Cloud Connector sufficient to connect the SAP BTP to an SAP ABAP back end or is SAP BTP Integration needed?

It depends on the scenario: For pure point-to-point connectivity to call on-premise functionality like BAPIs, RFCs, OData services, and so on, that are exposed via on-premise systems, the Cloud Connector might suffice.

However, if you require advanced functionality, for example, n-to-n connectivity as an integration hub, SAP BTP Integration – Process Integration is a more suitable solution. SAP BTP Integration can use the Cloud Connector as a communication channel.

How much bandwidth does the Cloud Connector consume?

The amount of bandwidth depends greatly on the application that is using the Cloud Connector tunnel. If the tunnel isn't currently used, but still connected, a few bytes per minute is used simply to keep the connection alive.

What happens to a response if there's a connection failure while a request is being processed?

The response is lost. The Cloud Connector only provides tunneling, it does not store and forward data when there are network issues.

Where should I install the Cloud Connector?

For productive instances, we recommend installing the Cloud Connector on a single purpose machine. This is relevant for [Security \[page 589\]](#). For more details on which network zones to choose for the Cloud Connector setup, see [Network Zones \[page 256\]](#).

How does a disaster recovery setup look like for the Cloud Connector?

There is no explicit implementation of a disaster recovery setup for the Cloud Connector. However, it is actually not needed.

Instead, make sure you have machines available in some other data center than the one in which your productive setup is running. Also, make sure you regularly generate a [Configuration Backup \[page 511\]](#).

If a disaster situation occurs, install the Cloud Connector again and restore the latest backup. Immediately after the restart that is required after restoring the backup, you are back to a running setup, as long as all the backend systems are reachable from the new location.

How many servers do I need to deploy the Cloud Connector?

We recommend that you use at least three servers, with the following purposes:

- Development
- Production master
- Production shadow

Note

Do not run the production master and the production shadow as VMs inside the same physical machine. Doing so removes the redundancy, which is needed to guarantee high availability. A QA (Quality Assurance) instance is a useful extension. For disaster recovery, you will need two additional instances: another master instance, and another shadow instance as a reserve for the disaster case.

What are the hardware requirements to deploy the Cloud Connector?

See: [Prerequisites \[page 239\]](#).

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 \[page 239\]](#).

Which processor architectures are supported by the Cloud Connector?

We currently support 64-bit operating systems running only on an x86-64 processor (also known as x64, x86_64 or AMD64), and for Linux also on the *PowerPC Little Endian* variant (also known as *ppc64le*).

See: [Prerequisites \[page 239\]](#).

Can I use the Cloud Connector without an ABAP back end?

Yes, you should be able to connect almost any system that supports the HTTP Protocol, to the SAP BTP, for example, Apache HTTP Server, Apache Tomcat, Microsoft IIS, or Nginx.

Can I authenticate with client certificates configured in SAP BTP destinations at HTTP services that are exposed via the Cloud Connector?

No, this is not possible. For client certificate authentication, an end-2-end TLS communication is required. This is not the case, because the Cloud Connector needs to inspect incoming requests in order to perform access control checks.

How can I do connection pooling for HTTP services that are exposed via the Cloud Connector?

The Cloud Connector itself does not perform connection pooling, but provides a 1-to-1 mapping for each logical connection received through the tunnel.

By this mapping, a new connection to the backend system is opened, and kept open until closed either by the backend or by the client on cloud side.

The actual connection pooling is defined by the application client on cloud side:

- If a connection is re-used in the client library, it is re-used on the Cloud Connector side as well.
- If it is closed immediately, also the mapped one on Cloud Connector side will be closed immediately.

Can I open two windows or tabs in a single browser instance to administrate the Cloud Connector?

No, this is not supported and may cause odd behavior on the different screens, in particular when trying to navigate through multiple subaccounts. If you like to open the administration UI twice, use two separate browser instances.

Does the Cloud Connector delete or modify HTTP headers?

Modifications of HTTP response headers are done if needed. In particular, Set-Cookie domains are adjusted according to the configured domain and host mappings. Also, in case of redirects, the location header will be adjusted according to the host mappings. Modifications of HTTP request headers are also done if needed, which is currently only the case for the *Host* header content. It will be replaced by the internal host, if the host mapping configuration is set up accordingly. The Cloud Connector will not delete any header that is sent by the cloud application. However, the Connectivity service will drop Connectivity service-specific headers, such as `SAP-Connectivity-Authentication` or `SAP-Connectivity-ConsumerAccount` so that those headers will neither reach the Cloud Connector nor the eventual backend.

Administration

Are there Audit Logs for changes in the Cloud Connector?

Yes, find more details here: [Manage Audit Logs \[page 576\]](#).

Is it possible to split authorization?

No, currently there is only one role that allows complete administration of the Cloud Connector.

Can I configure multiple administrative subaccounts?

Yes, to enable this, you must configure an LDAP server. See: [Use LDAP for User Administration \[page 521\]](#).

How can I reset the Cloud Connector's administrator password when not using LDAP for authentication?

Visit <https://tools.hana.ondemand.com/#cloud> to download the portable version of the Cloud Connector. Extract the `users.xml` file in the `config` directory to the `config` directory of your Cloud Connector installation, then restart the Cloud Connector.

This resets the password and user name to their default values.

You can manually edit the file; however, we strongly recommend that you use the `users.xml` file.

How do I create a backup of the Cloud Connector configuration?

Starting with Cloud Connector version 2.11, you can use a dedicated backup feature, either from the administration UI (see [Configuration Backup \[page 511\]](#)) or via REST API (see [Backup \[page 412\]](#)).

Can I create a backup of the complete installation?

Yes, you can create an archive file of the installation directory to create a full backup. Before you restore from a backup, note the following:

- If you restore the backup on a different host, the UI certificate will be invalidated.
- Before you restore the backup, you should perform a “normal” installation and then replace the files. This registers the Cloud Connector at your operating systems package manager.

Why do I need a user ID during configuration?

This user opens the tunnel and generates the certificates that are used for mutual trust later on.

The user is not part of the certificate that identifies the Cloud Connector.

In both the Cloud Connector UI and in the SAP BTP cockpit, this user ID appears as the one who performed the initial configuration (even though the user may have left the company).

What happens to a Cloud Connector connection if the user who created the tunnel leaves the company?

This does not affect the tunnel, even if you restart the Cloud Connector.

What do changes in major or minor version numbers mean?

The semantics of Cloud Connector versions are explained in detail [here](#) .

Does SAP provide support for older Cloud Connector versions?

SAP supports the latest 2 feature releases in parallel. For the latest feature release, the last 2 patch levels are supported. For the previous feature release, only its latest patch level is supported.

For more information on the Cloud Connector support strategy, see SAP Note [3302250](#) .

What is the difference between “subaccount name” and “subaccount user”?

SAP BTP customers can purchase subaccounts and deploy applications into these subaccounts.

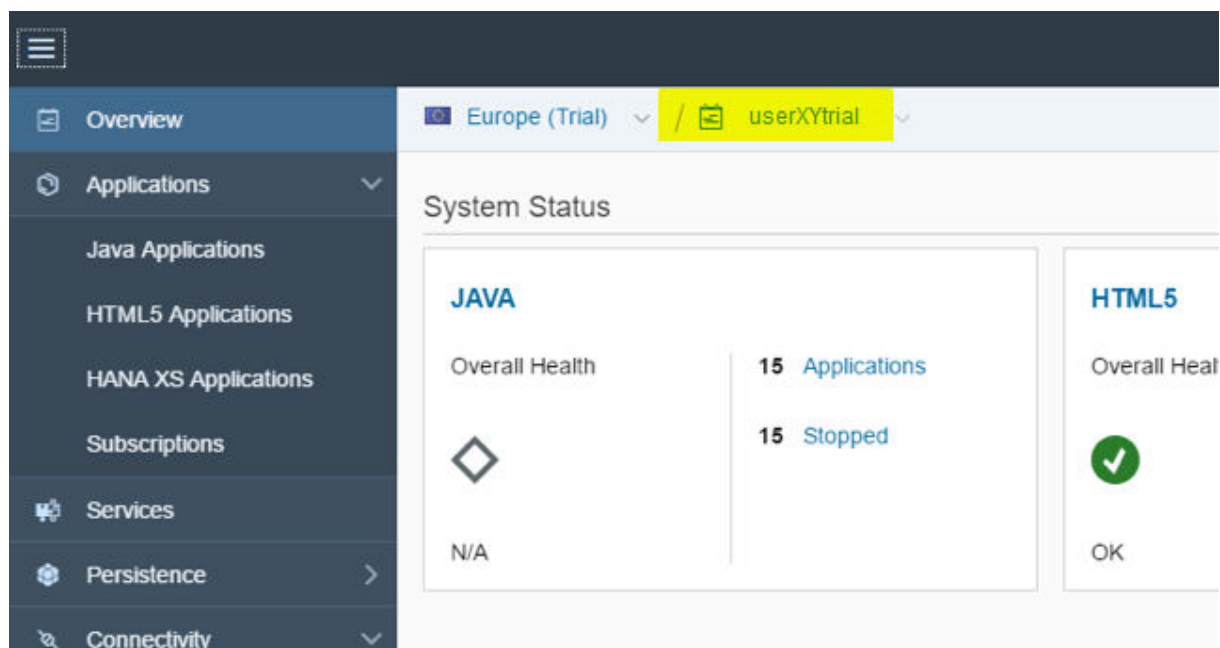
Additionally, there are users, who have a password and can log in to the cockpit and manage all subaccounts they have permission for.

- A single subaccount can be managed by multiple users, for example, your company may have several administrators.
- A single user can manage multiple subaccounts, for example, if you have multiple applications and want them (for isolation reasons) to be split over multiple subaccounts.

Find your account name by taking the following steps:

1. Open the SAP BTP cockpit.
2. Log in with your subaccount user.
3. You'll see the subaccount name in the top left section of the screen.

For trial users, the account name is typically your user name, followed by the suffix “trial”:



Features

Does the Cloud Connector work with the SAP BTP Cloud Foundry environment?

As of version 2.10, the Cloud Connector can establish a connection to regions based on the SAP BTP Cloud Foundry environment. Newer regions, however, require a Cloud Connector version 2.11 or higher.

Does the Cloud Connector work with SAP S/4HANA Cloud?

As of version 2.10, the Cloud Connector offers a Service Channel to S/4HANA Cloud instances, given that they are associated with the respective SAP BTP subaccount. For more information, see [Using Service Channels \[page 491\]](#).

Also supported as of version 2.10: S/4HANA Cloud communication scenarios invoking HTTP services or remote-enabled function modules (RFMs) in on-premise ABAP systems.

Does the Cloud Connector work with the SAP BTP ABAP environment?

As of version 2.11, the Cloud Connector supports communication from and to the SAP BTP ABAP environment, when using the **Neo** Connectivity service. Using the **Cloud Foundry** Connectivity service requires a Cloud Connector version 2.12.3 or higher.

How do I bind multiple Cloud Connectors to one SAP BTP subaccount?

As of version 2.9, you can connect multiple Cloud Connectors to a single subaccount. This lets you assign multiple separate corporate network segments.

Those Cloud Connectors are distinguishable based on the location ID, which you must provide to the destination configuration on the cloud side.

Note

During an upgrade, location IDs provided in earlier versions of the Cloud Connector are dropped to ensure that running scenarios are not disturbed.

Is WebSocket communication through the Cloud Connector supported?

Yes, this is possible as of version 2.12.

Is there any plan to add traffic management functionality in Cloud Connector?

No, this functionality is not currently planned.

Can I use the Cloud Connector from cloud to on-premise for any protocol?

As of version 2.10, you can use the TCP channel of the Cloud Connector, if the client supports a SOCKS5 proxy to establish the connection. However, only the HTTP and RFC protocols currently provide an additional level of access control by checking invoked resources.

Can I use the Cloud Connector from on-premise to cloud for any protocol?

This is possible only for a limited set of protocols. You can use the Cloud Connector as a JDBC or ODBC proxy to access the HANA DB instance within your SAP BTP Neo subaccount (service channel). This is sometimes referred to as “HANA protocol”. Also, there are service channels for SSH access to SAP BTP Neo virtual machines, and for RFC access to ABAP cloud systems. All of these service channels provide access to endpoints that are not visible in the Internet.

For HTTP, the endpoints that could be addressed are visible in the Internet. Therefore, you can simply use your normal network infrastructure that is prepared for accessing HTTPS endpoints in the Internet anyway.

Can I check the communication of the service channel?

No, the audit log monitors access only from SAP BTP to on-premise systems.

Troubleshooting

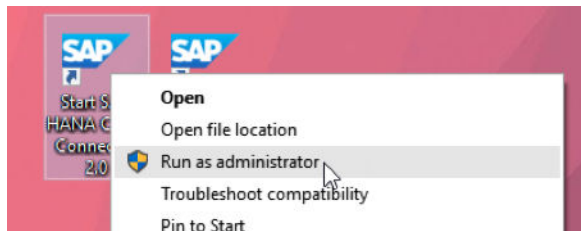
How do I fix the “Could not open Service Manager” error message?

You are probably seeing this error message due to missing administrator privileges. Right-click the cloud connector shortcut and select Run as administrator.

If you don't have administrator privileges on your machine you can use the portable variant of the Cloud Connector.

Note

The portable variants of the Cloud Connector are meant for nonproductive scenarios only.



How do I set JAVA_HOME and PATH correctly?

JAVA_HOME must point to the installation directory of your SAP JVM or JDK while PATH must contain the *bin* folder inside the installation directory of your SAP JVM or JDK. This is relevant in particular for the portable versions. The installers will also detect JDKs in other places.

How do I use the various .bat-Batch or .sh-Shell script tools in the Cloud Connector directory?

Open a command line prompt with administrator privileges or with sufficient user privileges to read and write files in the Cloud Connector directory. Then, make sure the environment variable JAVA_HOME is set to the installation directory of the JDK used by the Cloud Connector.

Afterwards, switch to the Cloud Connector directory and call the appropriate batch or shell script tools via `<toolname>.bat` or `./<toolname>.sh`. If the respective tool script requires further input parameters, its usage syntax will be written to the console.

When I try to open the Cloud Connector UI, Google Chrome opens a Save as dialog, Firefox displays some cryptic signs, and Internet Explorer shows a blank page, how do I fix this?

This happens when you try to access the Cloud Connector over HTTP instead of HTTPS. HTTP is the default protocol for most browsers.

Adding “https://” to the beginning of your URL should fix the problem. For localhost, you can use `https://localhost:8443/`.

1.6 Connectivity via Reverse Proxy

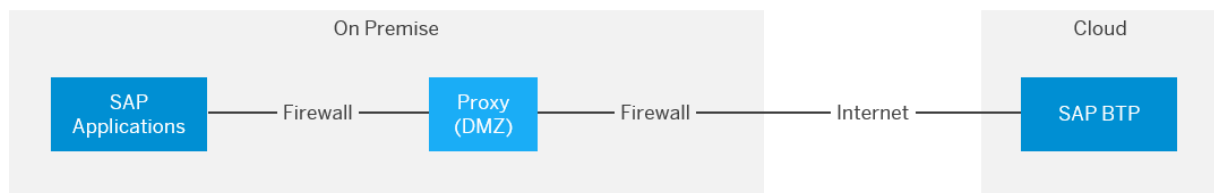
The text discusses the use of a reverse proxy as an alternative approach to connect on-premise services to SAP BTP. While it allows for reuse of existing network infrastructure, it exposes services to potential attacks and requires significant involvement from your IT department. The Cloud Connector is recommended as a more secure and efficient solution, providing TLS tunneling and fine-grained access control.

An alternative approach compared to the TLS tunnel solution that is provided by the Cloud Connector is to expose on-premise services and applications via a reverse proxy to the Internet. This method typically uses a reverse proxy setup in a customer's "demilitarized zone" (DMZ) subnetwork. The reverse proxy setup does the following:

- Acts as a mediator between SAP BTP and the on-premise services
- Provides the services of an Application Delivery Controller (ADC) to, for example, encrypt, filter, route, or check inbound traffic

The figure below shows the minimal overall network topology of this approach.

On-premise services that are accessible via a reverse proxy are callable from SAP BTP like other HTTP services available on the Internet. When you use destinations to call those services, make sure the configuration of the `ProxyType` parameter is set to **Internet**.



Advantages

Depending on your scenario, you may benefit from the reverse proxy:

- Network infrastructure (such as a reverse proxy and ADC services): since it already exists in your network landscape, you can reuse it to connect to SAP BTP. There's no need to set up and operate new components on your (customer) side.
- A reverse proxy is independent of the cloud solution you are using.
- It acts as single entry point to your corporate network.

Disadvantages

- The reverse proxy approach leaves exposed services generally accessible via the Internet. This makes them vulnerable to attacks from anywhere in the world. In particular, **Denial-of-Service attacks** are possible and difficult to protect against. To prevent attacks of this type and others, you must implement the highest security in the DMZ and reverse proxy. For the productive deployment of a hybrid cloud/on-

premise application, this approach usually requires intense involvement of the customer's IT department and a longer period of implementation.

- If the reverse proxy allows filtering, or restricts accepted source IP addresses, you can set only one IP address to be used for all SAP BTP outbound communications.
A reverse proxy does not exclusively restrict the access to cloud applications belonging to a customer, although it does filter any callers that are not running on the cloud. Basically, any application running on the cloud would pass this filter.
- The SAP-proprietary RFC protocol is supported only if WebSocket RFC can be used for communication with the ABAP system. WebSocket RFC is available as of S/4HANA release 1909. A cloud application cannot call older on-premise ABAP systems directly without using application proxies on top of ABAP in between.
- No easy support of principal propagation authentication, which lets you forward the cloud user identity to on-premise systems.
- You cannot implement projects close to your line of business (LoB).

Note


Using the Cloud Connector mitigates all of these issues. As it establishes the TLS tunnel to SAP BTP using a reverse invoke approach, there is no need to configure the DMZ or external firewall of a customer network for inbound traffic. Attacks from the Internet are not possible. With its simple setup and fine-grained access control of exposed systems and resources, the Cloud Connector allows a high level of security and fast productive implementation of hybrid applications. It also supports multiple application protocols, such as HTTP and RFC.

1.7 Connectivity Support

Support information for SAP BTP Connectivity and the Cloud Connector.

Troubleshooting

Locate the problem or error you have encountered and follow the recommended steps:

- [Frequently Asked Questions \[page 604\]](#) (Cloud Connector)
- [Administration \[page 516\]](#) (Cloud Connector)
- [Cloud Connectivity: Guided Answers](#) 
- [Getting Support](#) (SAP Support Portal, SAP BTP community)

SAP Support Information

If you cannot find a solution to your issue, collect and provide the following specific, issue-relevant information to SAP Support:



- The Java EE code that throws an error (if any)
- A screenshot of the error message for the failed operation or the error message from the `HttpResponse` body
- Access credentials for your cloud or on-premise location

You can submit this information by creating a customer ticket in the SAP CSS system using the following components:

Component	Purpose
Connectivity Service	
BC-CP-CON	For <i>cloud-side</i> issues with cloud to on-premise connectivity, where: <ul style="list-style-type: none">• The environment is unknown or• The issue is not related to a specific environment
BC-CP-CON-CF	For <i>cloud-side</i> issues with cloud to on-premise connectivity in the SAP BTP Cloud Foundry environment .
BC-CP-CON-S4HC	For <i>cloud-side</i> issues with cloud to on-premise connectivity in an S/4HANA Cloud system.
BC-CP-CON-K8S-PROXY	For <i>cloud-side</i> issues with cloud to on-premise connectivity in a Kubernetes cluster (or Kubernetes-based product), using the connectivity proxy software component.
BC-CP-CON-ABAP	For <i>cloud-side</i> issues with cloud to on-premise connectivity in the SAP BTP ABAP environment .
BC-NEO-CON	For <i>cloud-side</i> issues with cloud to on-premise connectivity in the SAP BTP Neo environment .
Destinations	
BC-CP-DEST	For issues with destination configurations , where: <ul style="list-style-type: none">• The environment is unknown or• The issue is not related to a specific environment
BC-CP-DEST-CF	For general issues with the Destination service in the SAP BTP Cloud Foundry environment , like: <ul style="list-style-type: none">• REST API• Instance creation, etc.
BC-CP-DEST-CF-CLIBS	For client library issues with the Destination service in the SAP BTP Cloud Foundry environment .
BC-CP-DEST-CF-TOOLS	For issues with the management of destination configurations via tools like the SAP BTP cockpit (Cloud Foundry environment).

Component	Purpose
BC-CP-DEST-NEO	For issues with destination configurations or: <ul style="list-style-type: none"> • Management tools • Client libraries, etc. related to destinations in the SAP BTP Neo environment .
Cloud Connector	
BC-MID-SCC	For connectivity issues related to installing and configuring the Cloud Connector , configuring tunnels, connections, and so on.

If you experience a more serious issue that cannot be resolved using only traces and logs, SAP Support may request access to the Cloud Connector. Follow the instructions in these SAP notes:

- To provide access to the Administration UI via a browser, see [592085](#) .
- To provide SSH access to the operating system of the Linux machine on which the connector is installed, see [1275351](#) .

Related Information

[Release and Maintenance Strategy \[page 617\]](#)

1.7.1 Release and Maintenance Strategy

Find information about SAP BTP Connectivity releases, versioning and upgrades.

Release Cycles

Updates of the Connectivity service are published as required, within the regular, bi-weekly SAP BTP release cycle.

New releases of the Cloud Connector are published when new features or important bug fixes are delivered, available on the [Cloud Tools](#) page.

Cloud Connector Versions

Cloud Connector versions follow the <major> . <minor> . <micro> versioning schema. The Cloud Connector stays fully compatible within a major version. Within a minor version, the Cloud Connector will stay with the same feature set. Higher minor versions usually support additional features compared to lower minor versions. Micro versions generally consist of patches to a <major> . <minor> version to deliver bug fixes.

For each supported major version of the Cloud Connector, only one <major> . <minor> . <micro> version will be provided and supported on the Cloud Tools page. This means that users must upgrade their existing Cloud Connectors to get a patch for a bug or to make use of new features.

For detailed support strategy information, check SAP note [3302250](#) .

Cloud Connector Upgrade

New versions of the Cloud Connector are announced in the [Release Notes](#) of SAP BTP. We recommend that Cloud Connector administrators regularly check the release notes for Cloud Connector updates. New versions of the Cloud Connector can be applied by using the Cloud Connector upgrade capabilities. For more information, see [Upgrade \[page 600\]](#).

Note

We recommend that you first apply upgrades in a test landscape to validate that the running applications are working.



There are no manual user actions required in the Cloud Connector when the SAP BTP is updated.

Important Disclaimers and Legal Information

Hyperlinks

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

About the icons:

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

Videos Hosted on External Platforms

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

Beta and Other Experimental Features

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

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

Example Code

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

Bias-Free Language

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

© 2024 SAP SE or an SAP affiliate company. All rights reserved.

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

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

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

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

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