Convenient Interaction with Data Space Integration (Beta)

Integration Package

Public

Table of Contents

Disclaimer

No part of the SAP Licensed End-User Documentation 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 in the SAP Licensed End-User Documentation 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.

SAP and other SAP products and services mentioned in the Licensed End-User Documentation 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 Trademark Information on SAP.com for additional trademark information and notices.

Introduction

This is the documentation for the Cloud Integration (CI) package “Convenient Interaction with Data Space Integration (Beta)”.

The integration package is used for an easier and more accessible way of interacting with the Data Space Integration capability of the SAP Integration Suite.

The integration package is available on SAP Business Accelerator Hub here.

Note that the EDC management API that this integration package uses is still in an experimental beta stage. This is why this package is marked as beta, too.

Used APIs

Data Space Integration

Data Space Integration is a new capability of SAP Integration Suite and enables convenient, secure, and self-sovereign data exchange in data spaces, such as Catena-X.

The current integration package supports Data Space Integration version 1.2.x on the consumer side (based on FOSS Eclipse Dataspace Components 0.7.2 and Tractus-X EDC 0.7.6). The following resources can be used to get information about the Data Space Integration API:

- CX-0018 Dataspace Connectivity 3.1.0
- Data Space Integration API 1.2.x

Known Issues

Currently, there are no known issues.

Restrictions

As this is a beta version of this integration package, it is not intended to use this for productive data.

Prerequisites

The following products and components are prerequisite for the integration package:

- SAP Integration Suite, with capabilities:
    - Cloud Integration
    - Data Space Integration

Overview

The following integration flow is used to negotiate and access the asset:

- Negotiate And Access Asset from Asset Provider

Deployment

Due to security and separation of concerns it is recommended to define separate DSI company policies for each use case. As company policies are bound to the client ID of the use case specific DSI service key which is configured in the integration flow, we must instantiate the integration package and the integration flow for each use case separately. Additionally the integration flow has to be deployed on a use case specific endpoint.

Request and Response Payloads

These are the properties of the request payload; note that the examples are for the Business Partner Data Management use case:

Property	Description	Example
counterPartyId	The BPNL of the asset provider	BPNL1234567890ZZ
counterPartyAddress	The data space protocol API of of the asset provider.	https://edc.provider.com/api/v1/dsp
catalogFilterProperties	A list of key value pairs to select the offer from the catalog.	
dct:type[1]	The mandatory asset type according to CX-0018.	BPDMGate (see CX-0074)
dct:subject[2]	The optional asset subject.	FullAccessGateInputForSharingMember (see CX-0074)
cx-common:version[3]	The mandatory asset version according to CX-0018.	6.0 (see CX-0074)
httpDataPlaneRequest	The sub object for the actual data plane request to the EDC of the asset provider.	
httpMethod	The HTTP method according to RFC 2616.	POST

[1] Note that the currently used JSON key by the BPDM use case is "https://purl.org/dc/terms/type"

[2] Note that the currently used JSON key by the BPDM use case is "https://purl.org/dc/terms/subject"

[3] Note that the currently used JSON key by the BPDM use case is "https://w3id.org/catenax/ontology/common/version"

Property	Description	Example
absolutePath	The absolute path to the API behind the asset, incl. query parameters.	/input/changelog/search?page=0&size=100
httpBody	The HTTP body sub object to send to the API behind the asset.	
contentType	The optional content type for the body according to RFC 2616.	application/json
characterEncoding	The character encoding of the body according to ISO/IEC 10646:2020.	UTF-8
bodyAsBytes	The base64 encoded body.	

These are the properties of the response payload:

Property	Description	Example
statusCode	The HTTP status code according to RFC 2616.	200
statusText	The HTTP status text according to RFC 2616.	OK
errorMessage	The error message in case of a status code with 4xx or 5xx.	
httpBody	The http body sub object returned from the API behind the asset.	
contentType	The content type which the consumer can expect in the body according to RFC 2616.	application/json
characterEncoding	The character encoding of the body according to ISO/IEC 10646:2020.	UTF-8
bodyAsBytes	The base64 encoded body.	

The following HTTP status codes are returned in case of an error:

- 404 - When no offer is found, the API returns 'The offer you were looking for does not exist in the catalog'
- 400 - When too many offers are found, the API returns 'Your request was invalid. Too many offers match the selected filter criteria

For example payloads see Example Payloads.

Process

Negotiate the Asset

With the negotiate and access asset request the asset provider and its data space protocol catalog endpoint are uniquely identified (counterPartyId and counterPartyAddress). The control plane is used to get the right offer from the catalog of the asset provider based on the filter properties (catalogFilterProperties) send with the request. Note that the offer is additionally filtered based on the consumer / provider usage policies (in Data Space Integration) and the consumer BPNL, if an access policy for the consumer is in place at the

provider. Note also that the list of filter properties is use case specific. If not exactly one offer is returned for these criteria the integration flow will fail and return an error message.

Next, it is checked based on the asset id from the offer and the provider BPNL, whether there is a valid endpoint data reference (EDR) with a short-lived token cached. If this is the case, a data exchange contract exists and does not need to be negotiated. If a cached EDR is not returned, the next step is to negotiate the data exchange contract based on the ID, the provider usage policy and the data space protocol negotiation endpoint (distribution) of the offer. Once negotiation is finalized and the agreement has been returned, it is again checked, whether there is now a valid endpoint data reference (EDR) with a short-lived token cached.

Access the Asset

As a next step the short-lived token is used to authenticate and communicate with the API behind the asset (data plane) using the HTTP parameters from the request (httpMethod, absolutePath, httpBody). Once processing on provider side has finished, the response is returned to the caller of the integration flow (statusCode and httpBody).

Configuration Steps

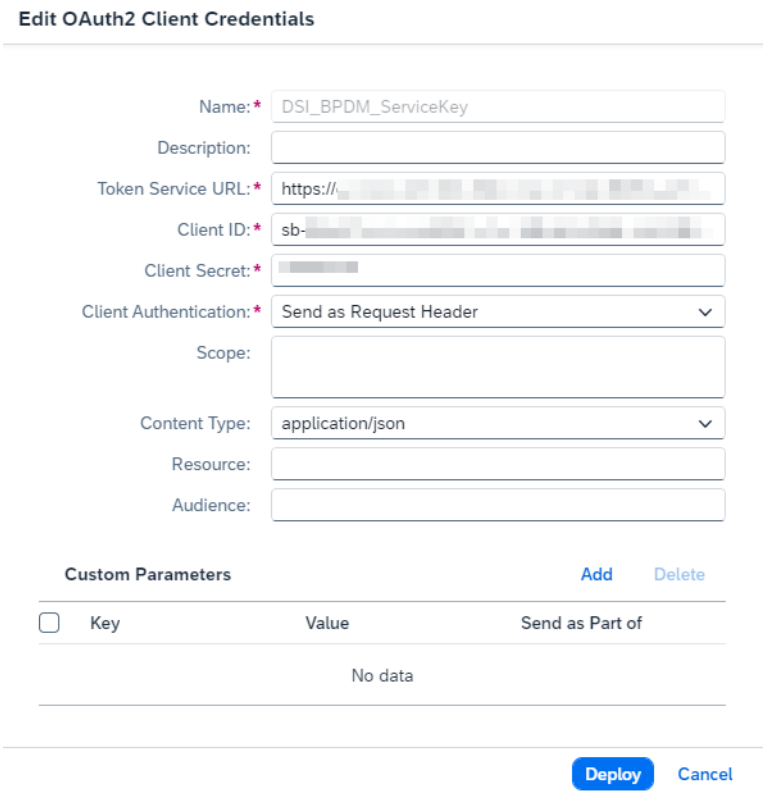
The following steps describe, how to setup the integration between an SAP application and Data Space Integration.

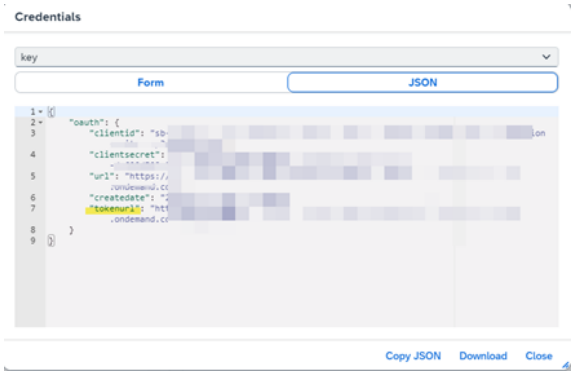
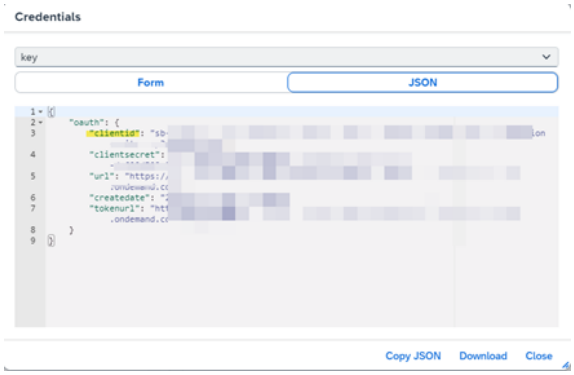
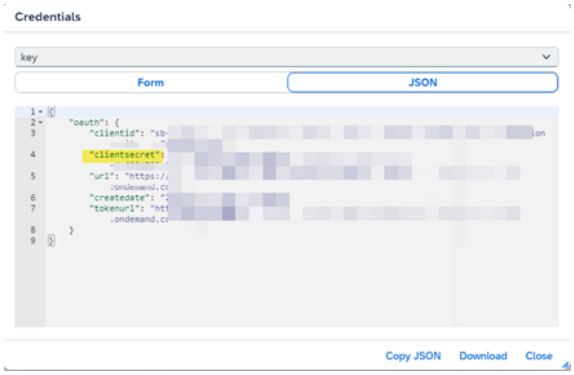
Data Space Integration

Information about the initial setup of Data Space Integration can be found here.

Accessing DSI from CI

- In SAP BTP Cockpit, create a dedicated service key in the corresponding subaccount (see Using APIs To Work With Data Space Integration | SAP Help Portal and Creating Service Instance and Service Key for Inbound Authentication | SAP Help Portal) of type “OAuth2 ClientId/Secret” for a new or existing service instance “Data Space Integration API access”, the plan “api” and at least role “AuthGroup_DataspaceConsumer”; to create service instance and keys you need the following prerequisites:
    - Subaccount admin role
    - Cloud Foundry organization member
    - Assignment to Cloud Foundry space
- In SAP Integration Suite under “Security Material”, create a new security material for CI (see Managing Security Material | SAP Help Portal) of type “OAuth2 Client Credentials” with a name, containing the use case name.



Parameter Name	Parameter Value
Name	e.g. DSI_{UseCaseName}_ServiceKey; replace {UseCaseName} with the name of your use case.
Token Service URL	 Paste here value from parameter “tokenurl” from the service key configured in the step above
Client ID	 Paste here value from parameter “clientid” from the service key configured in the step above
Client Secret	 Paste here value from parameter “clientsecret” from the service key configured in the step above
Client Authentication	Send as Request Header
Content Type	application/json

Use this security material for authenticating the “Negotiate and Access Asset from Asset Provider” integration flow for communication with Data Space Integration.

Add and Assign Company Policy

Add new company policy corresponding to your use case (see Create and Edit Company Policies | SAP Help Portal). You can reuse an existing company policy template. Assign the client id from the step before to the company policy (see Assign Company Policies | SAP Help Portal).

Cloud Integration

Information about the initial setup of Cloud Integration can be found here.

Accessing the Integration Flow on CI

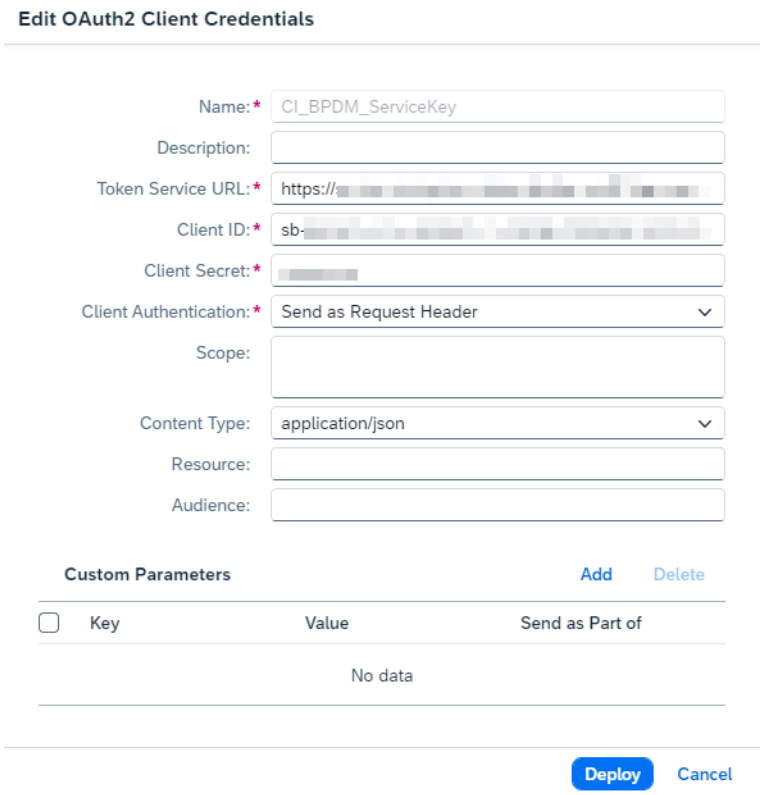
- In SAP BTP Cockpit, create a dedicated service key in the corresponding subaccount (see Creating Service Instance and Service Key for Inbound Authentication | SAP Help Portal) of type “OAuth2 ClientId/Secret” for a new or existing service instance “Process Integration Runtime”, the plan “integration-flow” and at least role “ESBMessaging.send”; to create service instance and keys you need the following prerequisites:
    - Subaccount admin role
    - Cloud Foundry organization member
    - Assignment to Cloud Foundry space

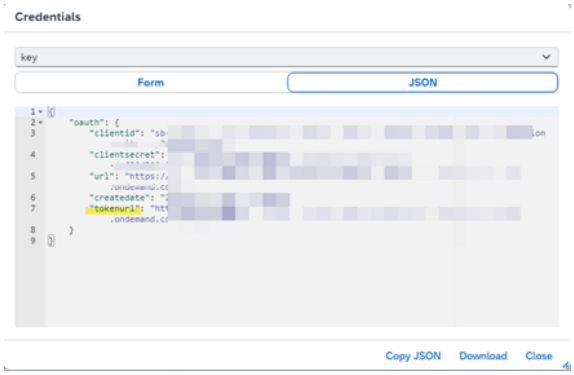
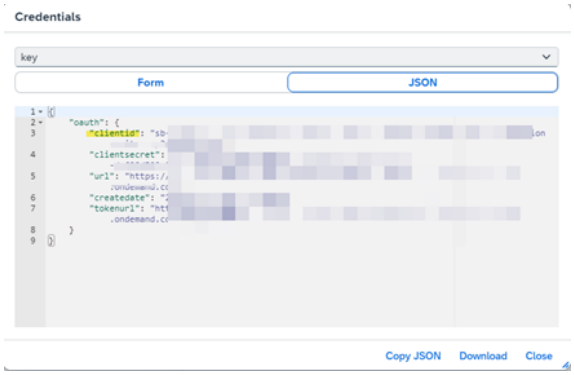
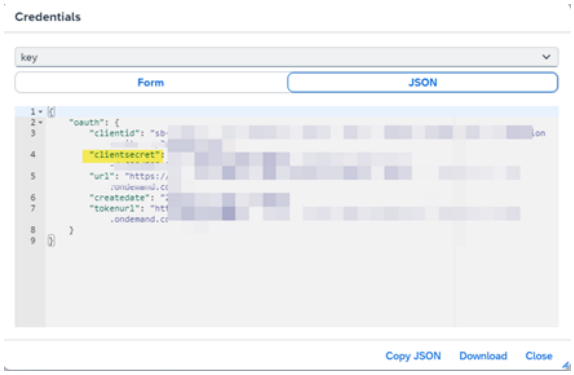
Use this service key for authenticating your application for communication with the “Negotiate and Access Asset from Asset Provider” integration flow.

Communication between Integration Flows on CI

If the integration flow is called by another integration flow, you need to configure the following:

- In SAP Integration Suite under “Security Material”, create a new security material for CI (see Managing Security Material | SAP Help Portal) of type “OAuth2 Client Credentials” with a name, containing the use case name.



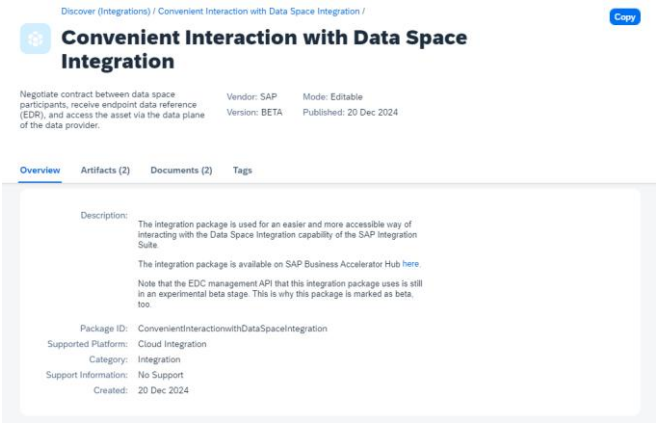
Parameter Name	Parameter Value
Name	e.g. CI_{UseCaseName}ServiceKey; replace {UseCaseName} with the name of your use case.
Token Service URL	 Paste here value from parameter “tokenurl” from the service key configured in the step above
Client ID	 Paste here value from parameter “clientid” from the service key configured in the step above
Client Secret	 Paste here value from parameter “clientsecret” from the service key configured in the step above
Client Authentication	Send as Request Header
Content Type	application/json

Use this security material for authenticating your integration flow for communication with the “Negotiate and Access Asset from Asset Provider” integration flow.

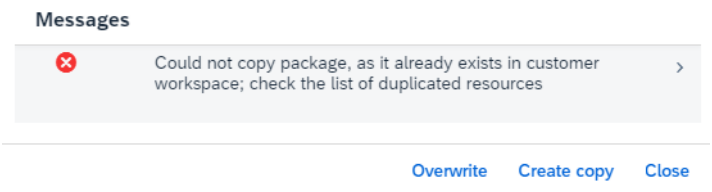
Integration Package

1. Connect to SAP Integration Suite
2. Go to **Discover (Integrations)** view.

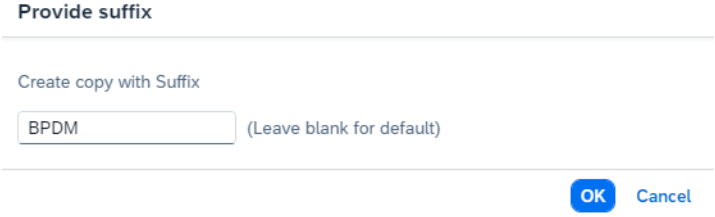
3. Search for the package **Convenient Interaction with Data Space Integration (Beta)**.
4. Select the package and copy it to your **Design** workspace by clicking the **Copy** button in the right corner.
5. Select the package and copy it to your **Design** workspace a second time by clicking the **Copy** button in the right corner.



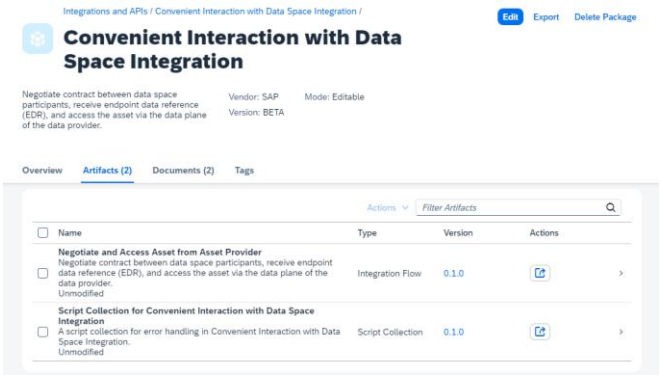
6. You must instantiate the integration package one time for each use case for which you want to use Data Space Integration by clicking on **Create copy** button in the following message.



7. Create the copy with the use case as suffix.



8. After copying go to **Design** (Integrations and APIs) open the package **Convenient Interaction with Data Space Integration (Beta)**

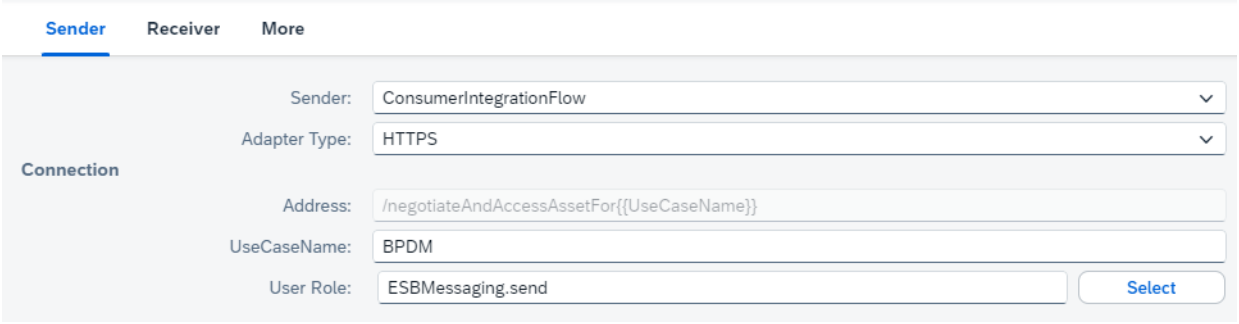


9. All the integration flows will be shown / listed under the **Artifacts** section of the page.
10. Change the name of the integration package and flow to include the use case name
11. Delete the integration package copy from step 4
12. Select the integration flow **Negotiate and Access Asset from Asset Provider** described in the table below and choose **Actions** button in the right and select **Configure**.

Negotiate and Access Asset from Asset Provider

Sender ConsumerIntegrationFlow

The ConsumerIntegrationFlow sender adapter exposes a hardcoded (but partially configurable) HTTPS endpoint "/negotiateAndAccessAssetFor{{UseCaseName}}" through which a calling external integration flow or application can negotiate and access an asset for a specific use case.



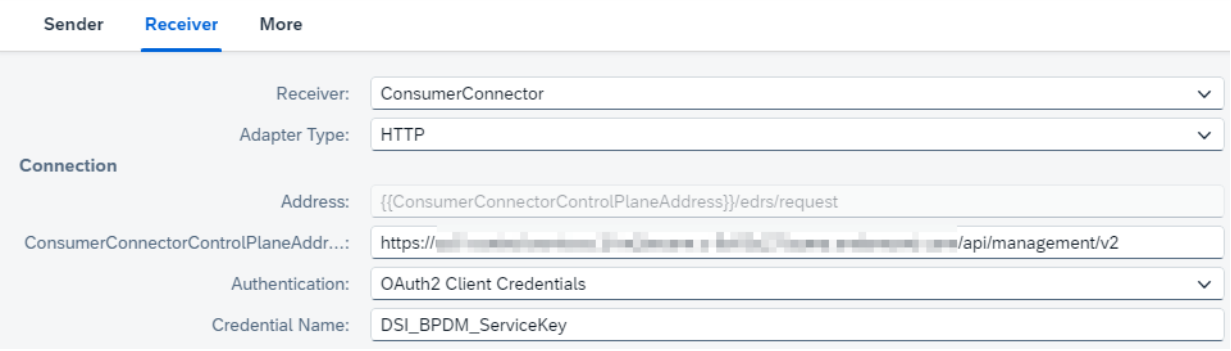
Parameter	Description
UseCaseName	The name of the use case that shares data via a data space. The name is appended to the HTTPS endpoint.
User Role	Choose **Select** to get a list of all available roles. You can use the role **ESBMessaging.send** .[4]

[4] It is a predefined role provided by SAP that authorizes a sender system to process messages on a tenant. However, using SAP BTP Cockpit, you can also define custom roles for the runtime node as well (we

Receiver ConsumerConnector

Via the ConsumerConnector receiver adapter the following messages are sent to the management API of DSI:

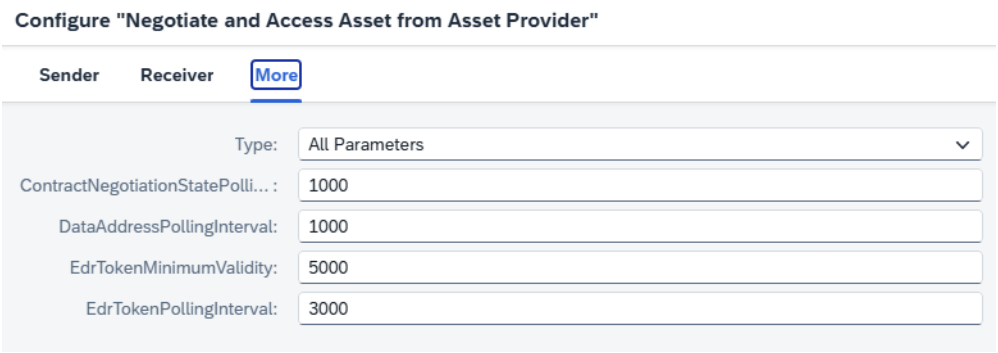
- Query Catalog
- Check EDR Cache
- Initiate EDR Negotiation
- Get Contract Negotiation State
- Get EDR Data Address



Parameter	Description
Adapter Type	For each call mentioned above there is one entry here. However, for all calls the same address and security material are used.
ConsumerConnectorControlPlaneAddress	The address of your DSI control plane management API, in the form <protocol>://<host>/api/management/v2.
Authentication	Choose **Select** to get a list of all available authentication types, recommended is **OAuth2 Client Credentials** which is provided by default.
Credential Name	The name of the security material, containing the user credentials used for Accessing DSI from CI.

recommend the creation of an individual sender role). When you choose **Select**, a selection of all custom roles defined that way is offered.

Additional Parameters



Parameter	Description
ContractNegotiationStatePollingInterval	Time interval in milliseconds for polling **Get Contract Negotiation State** for FINALIZED after **Initiate EDR Negotiation**. Default value is configured to 1000.
DataAddressPollingInterval	Time interval in milliseconds for polling **Get Data Address** for a valid EDR token after **Get EDR**. Default value is configured to 1000.
EdrTokenMinimumValidity	Minimum validity of an EDR token in milliseconds for being used in **Access Asset**. Default value is configured to 5000.
EdrTokenPollingInterval	Time interval in milliseconds for polling **Get EDR** after **Get Contract Negotiation State** is FINALIZED. Default value is configured to 3000.

Note that the maximum number of polling iterations is set to 20 in all polling processes. This means that the maximum time for polling is 20 times the polling interval.

After configuration you can deploy the integration flow. 

Other Artifacts

After configuration of the integration flows also deploy the other artifacts:

- Script Collection for Business Partner Data Management

Appendix

Example Payloads

Request

```
{
 "counterPartyId": "{{PROVIDER_BPNL}}",
 "counterPartyAddress": "{{PROVIDER_CONNECTOR_DATASPACE_API}}",
 "catalogFilterProperties": [
  {
   "name": "https://purl.org/dc/terms/type",
   "value": "cx-taxo:{{ASSET_TYPE}}"
  },
  {
   "name": "https://purl.org/dc/terms/subject",
   "value": "cx-taxo:{{ASSET_SUBJECT}}"
  },
  {
   "name": "https://w3id.org/catenax/ontology/common/version",
   "value": "{{ASSET_VERSION}}"
  }
 ],
 "httpDataPlaneRequest": {
  "httpMethod": "{{HTTP_METHOD}}",
  "absolutePath": "{{ABSOLUTE_PATH}}",
  "contentType": "{{CONTENT_TYPE}}",
  "httpBody": {
   "characterEncoding": "{{CHARACTER_ENCODING}}",
   "bodyAsBytes": "{{BASE64_ENCODED_BODY}}"
  }
 }
}
```

Response

```
{
 "statusCode": "{{STATUS_CODE}}",
 "statusText": "{{STATUS_TEXT}}",
 "errorMessage": "{{ERROR_MESSAGE}}",
 "contentType": "{{CONTENT_TYPE}}",
 "httpBody": {
  "characterEncoding": "{{CHARACTER_ENCODING}}",
  "bodyAsBytes": "{{BASE64_ENCODED_BODY}}"
```

```
  }
}
```