

## Azure Blob Storage connectivity to SAP Cloud Integration with help of SAP Private Link service



# Content

<b>1</b>	<i>Documentation</i> .....	<b>4</b>
1.1	Create a Service Instance for SAP PrivateLink Service .....	4
1.2	Configure and deploy Application Router .....	5
1.3	Use the Application Router proxy in SAP Cloud Integration iFlows .....	6
<b>2</b>	<i>Resources</i> .....	<b>10</b>



## 1 Documentation

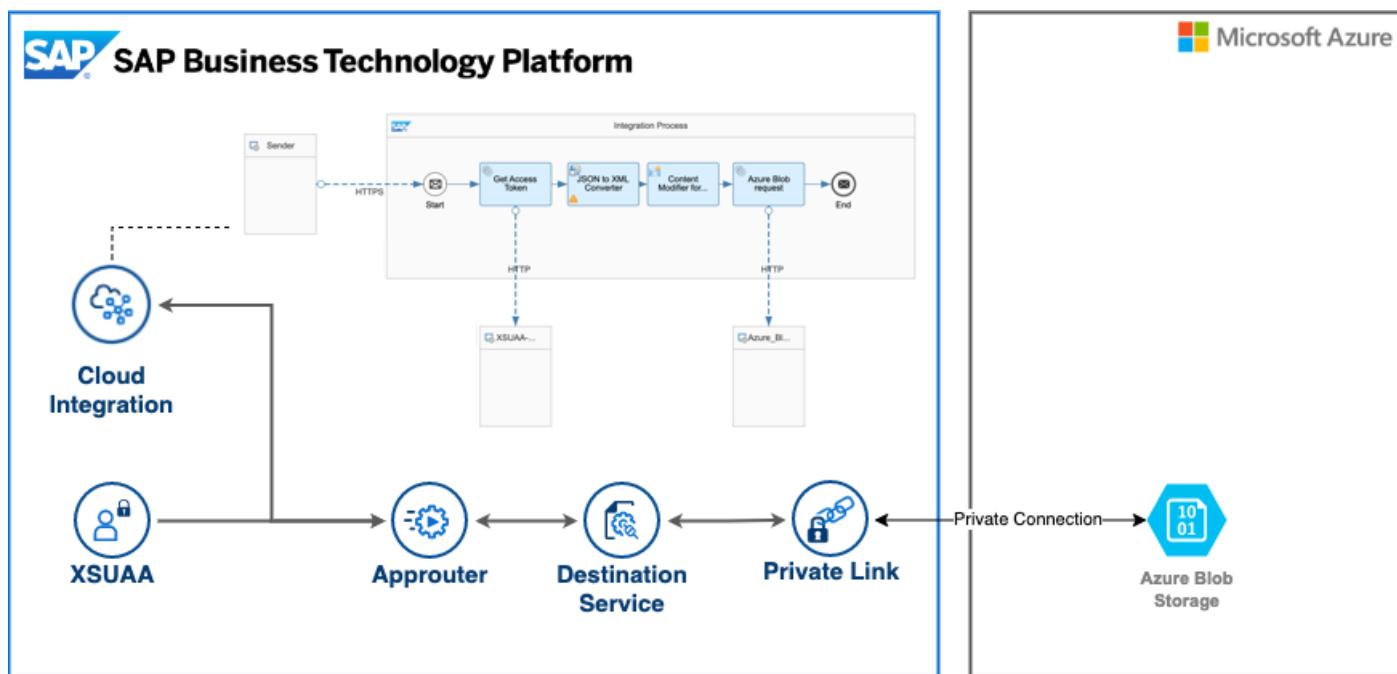
This guide illustrates the necessary steps for setting up Azure Blob Storage connectivity to SAP Cloud Integration with help of SAP Private Link service

The SAP Cloud Integration capability of SAP Integration Suite enables enterprises to connect different systems and applications in hybrid and cloud landscapes, that are developed and maintained on different technology stacks. These stacks, usually follow different security standards and requirements.

With the help of the SAP Private Link service, you can extend your hybrid integration scenarios to suit stricter security policies and communicate with your workloads on Microsoft Azure through private network connectivity. Azure Blob storage can now be linked to SAP Cloud Integration, enabling the easy exchange of massive amounts of unstructured data like images and documents. This integration provides a secure solution for businesses that need to transfer unstructured data between the two platforms.

The main idea of this architecture is to use the Application Router as a proxy for the private connectivity between Azure Blob Storage and SAP Cloud Integration. Detailed configuration steps you can find below.

Please note that for the moment, we cannot use SAP Private Link service directly from SAP Cloud Integration; nevertheless, we can bridge this gap with help of SAP's Application Router (approuter), which can play the role of a proxy between SAP Private Link service and SAP Cloud Integration, meanwhile product team is working to enable direct integration.



## 1.1 Create a Service Instance for SAP PrivateLink Service

You can either use the **CLI** or **BTP Cockpit** to create a service instance. Below you can find sample command.

Please note to replace the **resourceId** with your Azure Blob Storage resourceId

```
cf cs privatelink standard privatelink-blob -c '{"requestMessage":"Please approve blob connection","resourceId":"/subscriptions/xxxxxx/resourceGroups/tests-services/providers/Microsoft.Storage/storageAccounts/xxxxx","subResource":"blob"}'
```

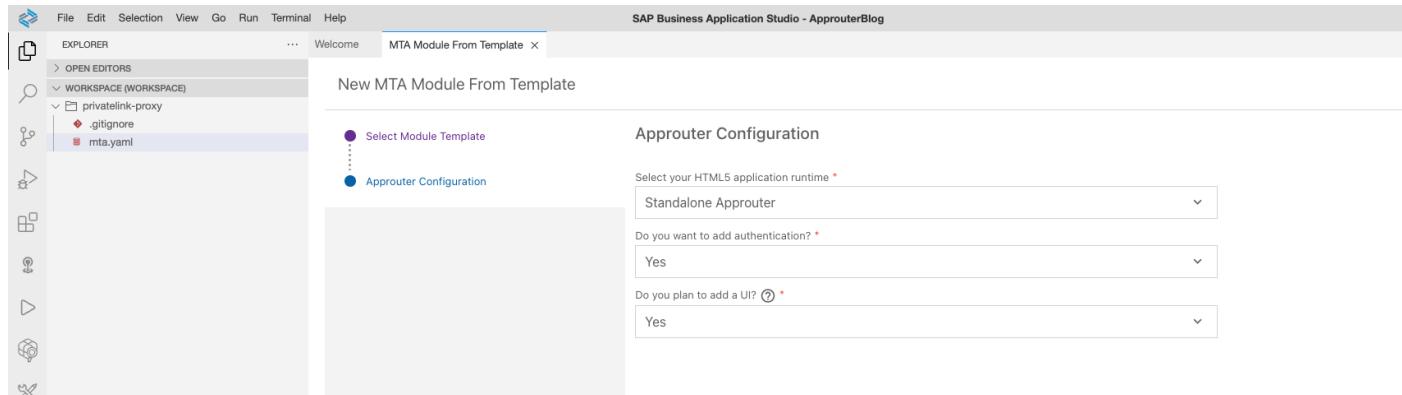
After initiation, remember to approve the connection request from **Azure Portal**.

## 1.2 Configure and deploy Application Router

You can clone this repository and use the provided sample approuter by using the IDE of your choice.

Another approach is using **SAP Business Application Studio (BAS)**, where you can take advantage of the provided templates for your Application Router configuration.

If you choose BAS, select **Standalone Approuter** as an option in the template wizard.



Define the route and the destination used for the SAP Private Link connectivity. This can be done in the xs-app.json file where **blob-approuter** is the destination configured for SAP Private Link connectivity in the target SAP BTP subaccount (see destination configuration below).

NOTE: Please change the **source** and **target** properties as required in your scenario (e.g. other path instead of /myfile)

```
{  
  "authenticationMethod": "route",  
  "routes": [  
    {  
      "source": "^/myfile/(.*)$",  
      "target": "/myfile/$1",  
      "destination": "blob-approuter",  
      "authenticationType": "xsuaa",  
      "csrfProtection": false  
    }  
  ]  
}
```

Destination Configuration					
Type	Name	Basic Properties			
HTTP	blob-approuter	Authentication	NoAuthentication	PrivateLink	https://testbtpapprouter.blob.core.windows.net

Destination Configuration

Name: * <input type="text" value="blob-approuter"/>	Additional Properties
Type: <input type="text" value="HTTP"/>	<input checked="" type="checkbox"/> Use default JDK truststore
Description: <input type="text"/>	
URL: * <input type="text" value="https://testbtpapprouter.blob.core.windows.net"/>	
Proxy Type: <input type="text" value="PrivateLink"/>	
Authentication: <input type="text" value="NoAuthentication"/>	

[Edit](#) [Clone](#) [Export](#) [Delete](#)

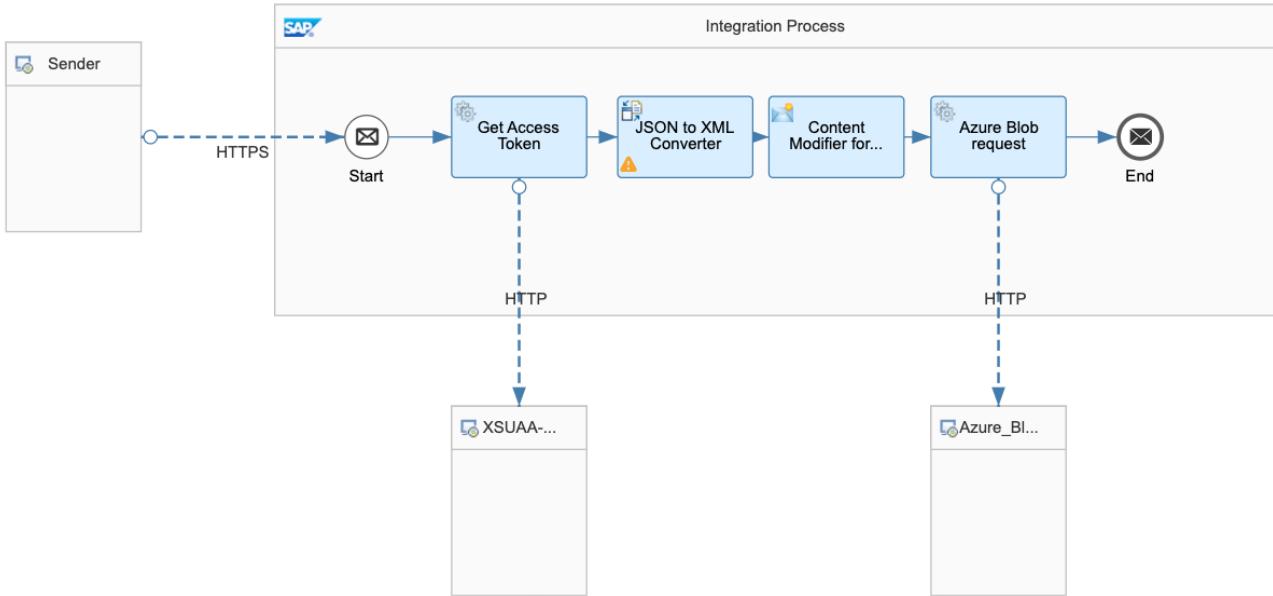
After setting up the route with destination and authentication, you can deploy the Application Router to your SAP BTP subaccount e.g. by

```
mbt build && cf deploy mta_archives/privateliink-proxy_0.0.1.mtar
```

Once the Application Router is up and running, it can be used by your integration flows to connect with the Azure Blob Storage.

## 1.3 Use the Application Router proxy in SAP Cloud Integration iFlows

All your standard integration patterns can stay the same, you need to add a couple of additional steps to use the approuter proxy with the private link connectivity to your Azure Blob Storage.



Following additional steps are required:

1. Get the access token from the XSUAA component of approuter. Please maintain the *client\_id* and *client\_secret* of XSUAA component as "User Credentials" in the **Security Material** of SAP Cloud Integration

**HTTP**

General Connection

**CONNECTION DETAILS**

Address: `https://...authentication.eu20.hana.ondemand.com/oauth/token`  
Query: `grant_type=client_credentials`  
Proxy Type: Internet  
Method: POST  
Authentication: Basic  
Credential Name: approuter-uc  
Timeout (in ms): 60000  
Throw Exception On Failure:

**HEADER DETAILS**

Request Headers: \*  
Response Headers: \*

2. Transform the response to XML (to add in a next step the custom header attribute)

**JSON To XML Converter**

General Processing

Use Namespace Mapping:

Namespace Mapping:

JSON Prefix	XML Namespace
root	

JSON Prefix Separator: Colon(:)  
Add XML Root Element:   
Name: root  
Namespace Mapping:

3. Modify the content by adding the header "x-approuter-authorization" (required by approuter) from the Authorization header value

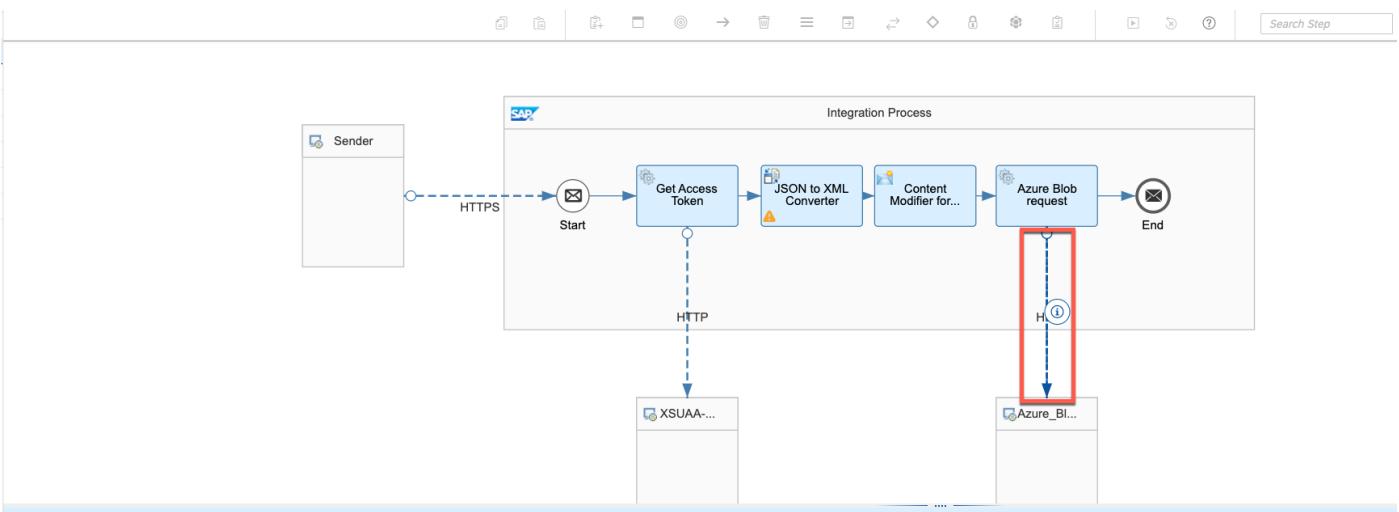
**Content Modifier**

- General Message Header Exchange Property Message Body

Headers:

Action	Name	Source Type	Source Value	Data Type	Default Value
Create	token	XPath	/root/access_token	java.lang.String	
Create	x-approuter-authorization	Expression	Bearer \${header.token}		

4. Make the final call to the approuter, which will route the request to Azure Blob Storage via Private Link connectivity  
The address correspondence to the Approuter URL



**HTTP**

- General Connection

CONNECTION DETAILS

Address:	https://[REDACTED].cfapps.eu21.hana.ondemand.com
Query:	sp=r
Proxy Type:	Internet
Method:	GET
Send Body:	<input type="checkbox"/>
Authentication:	None
Timeout (in ms):	60000
Throw Exception On Failure:	<input checked="" type="checkbox"/>

HEADER DETAILS

Request Headers:	x-approuter-authorization
Response Headers:	*

Having all these steps in place, you can deploy the integration flow and test it by calling your integration endpoint.

The example iFlow configuration you can get [here](#) and import to your SAP Cloud Integration tenant.

Go to your Cloud Integration cockpit and **import** the above-provided package.

The screenshot shows the SAP Cloud Integration cockpit. In the top right corner, there is a red box around the 'Import' button. Below the header, there is a search bar and a table titled 'Packages (1)'. The table has columns: Name, Mode, Version, Created By, Created Date, Description, and Action. One row is visible, showing a package named [REDACTED] created by maximilian.straaten@ondemand.com on Tue, 16 Aug 2022 15:48:16 GMT with the description 'Use Approuter to reach backend system via PrivateLink'.

Open the imported package

The screenshot shows the SAP Cloud Integration cockpit after the package has been imported. The 'Packages (2)' table now includes the imported package 'PrivateLinkProxy'. The table columns are identical to the previous screen. The 'PrivateLinkProxy' row is highlighted with a red box, showing it is Editable, Version 1.0.0, created on Tue, 16 Aug 2022 15:48:16 GMT, and has the same description: 'Use Approuter to reach backend system via PrivateLink'.

## Open the integration flow

Integrations / PrivateLinkProxy / PrivateLinkProxy

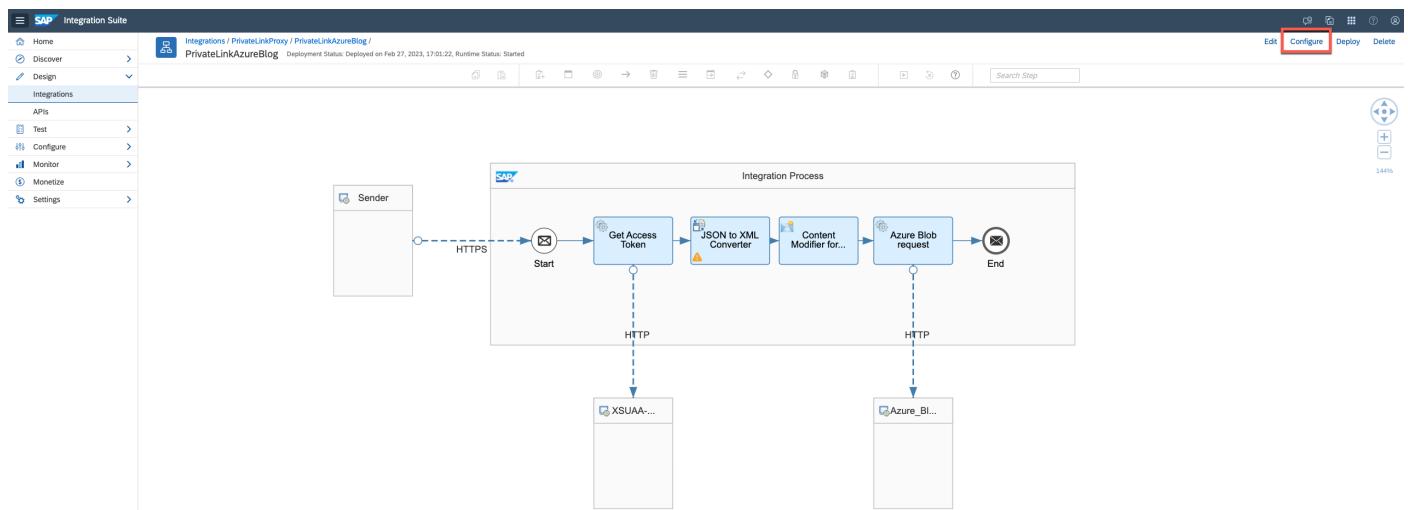
Use Approuter to reach backend system via PrivateLink

Vendor: Mode: Editable Version: 1.0.0

Overview Artifacts (1) Documents Tags Actions

Name	Type	Version	Actions
PrivateLinkAzureBlog	Integration Flow	1.0.30	
Modified			

Configure the parameters based on your tenant



Provide the XSUAA URL and Credential name from Security Material

Configure "PrivateLinkAzureBlog"

Receiver

Receiver: XSUAA-Approuter

Adapter Type: HTTP

Connection

Address:	https://<xsuaa-url>/oauth/token
Credential Name:	approuter-uc

Save Deploy Close

Provide the Approuter URL

Configure "PrivateLinkAzureBlog"

**Receiver**

Receiver: Azure\_Blob

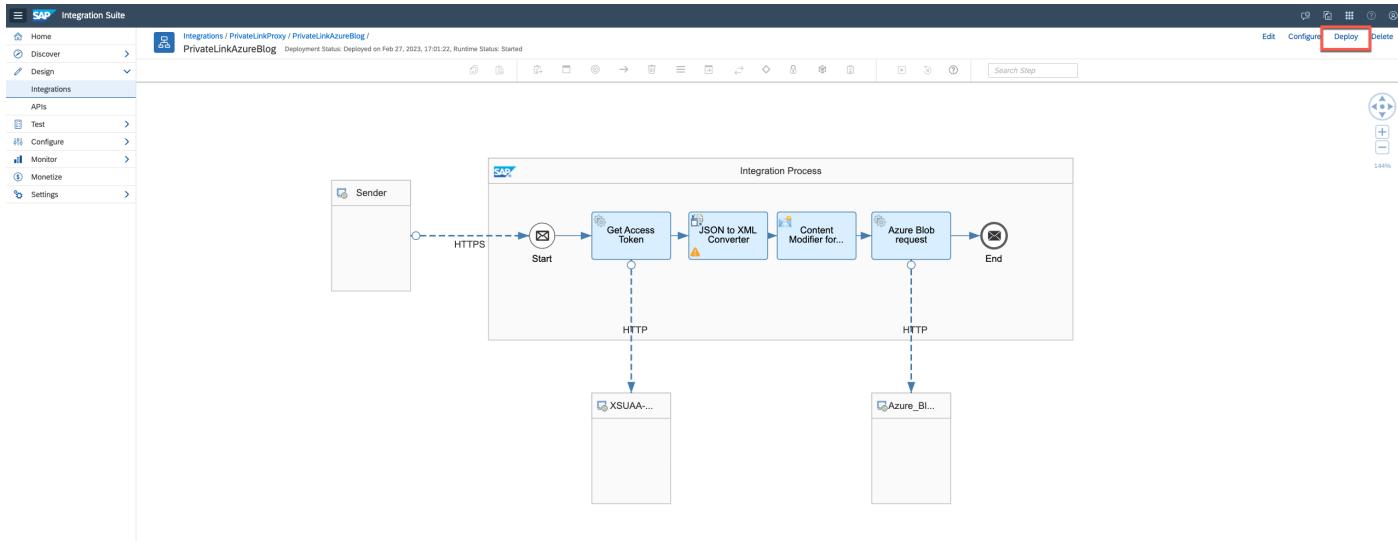
Adapter Type: HTTP

**Connection**

Address: https://

Save Deploy Close

## Deploy the iFlow



After deployment you can run your integration flow based on SAP Private Link connectivity.

## 2 Resources

[Private Link Service](#)