Integration Package Documentation: Figaf Azure DevOps Integration

Version: 1.0.1
Last Updated: 23.07.2025
Contact: Alexandru Florea (afl@figaf.com)

Contents

1. Package Overview

This integration package facilitates communication and process automation between the Figaf DevOps tools and Azure DevOps. It enables organizations to synchronize ticket and transport information, automating Azure issue updates based on Figaf events, and providing utilities for querying Azure data.

The package currently includes the following key integration flows (iFlows):

- **Azure_SearchIssues:** Provides an endpoint to search for Azure issues based on specified criteria, called by Figaf.
- **Azure_UpdateIssue:** Processes webhook events from Figaf to update or create corresponding Azure issues.
- **Azure_FigafUpdateTransport:** Allows Azure webhooks to trigger transport imports in Figaf.

This document provides details on the configuration and functionality of each iFlow.

2.Prerequisites

- SAP Integration Suite (CPI) tenant.
- Access to a Figaf instance with webhook capabilities.
- Access to a Azure DevOps instance with API access.
- Necessary authorizations in both Figaf and Azure for the integration user/API token.
- User Credentials artifact must be deployed on the CPI tenant The credential (ex: AZURE_ApiService_Credentials) , containing the Azure API token (e.g., Username: your_Azure_email, Password: your_Azure_api_token).

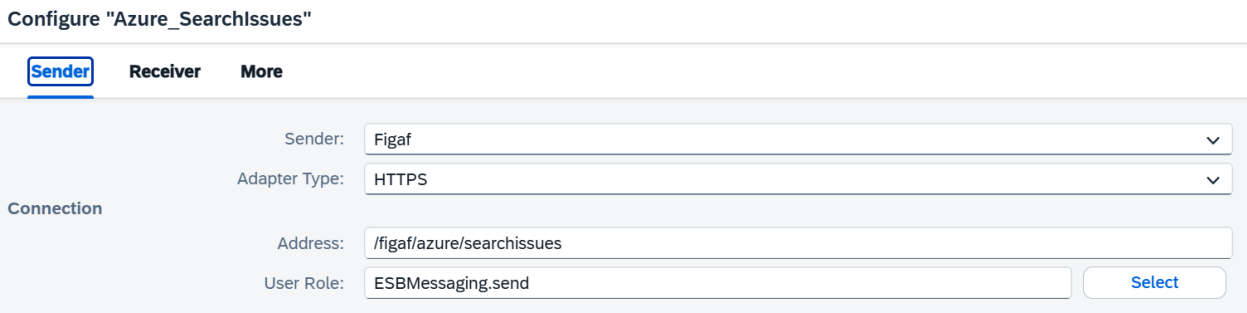
3. Integration Flows

3.1. Azure_SearchIssues

Provides an endpoint for Figaf (or other systems) to query Azure and retrieve essential field data (key, summary, description, status) for issues within a specified project.

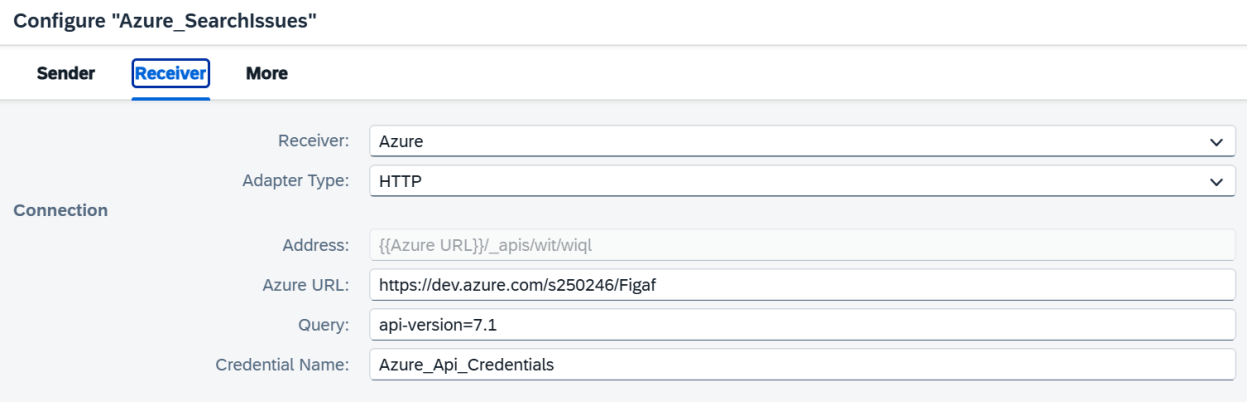
3.1.1. Configuration (from iFlow's "Configure" tab):

Sender Details:



- **Address:** This is the path appended to your Integration Suite tenant URL to call this iFlow.
- **User Role:** The calling system must authenticate with a user having this role.

Receiver Details (Connection to Azure):



- **Azure URL**: The base Azure path to your project.
- **Query**: Only used in this case for api version.
- **Credential Name**: The alias of the User Credential artifact in Integration Suite for Azure API authentication.

3.1.2. Using the IFlow

IFlow Trigger: HTTPS call, typically from Figaf.

Input: None.

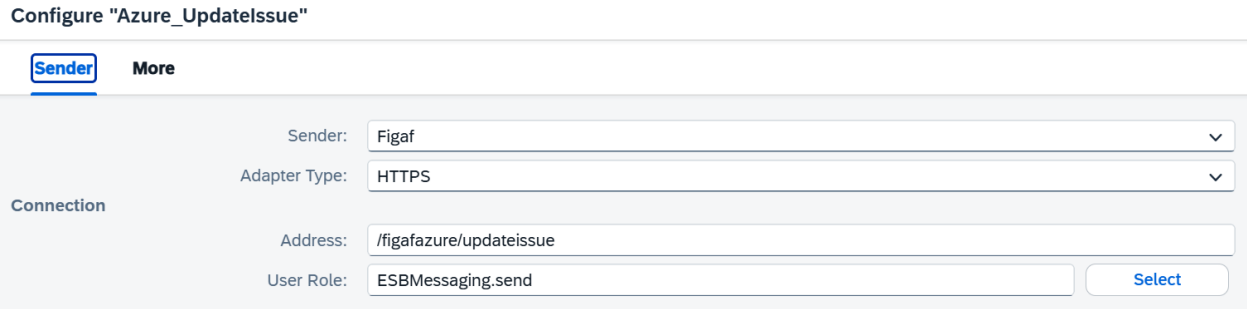
Output: JSON response from Azure containing the search results.


```
[
    {
        "ID": "3",
        "URL":
"https://dev.azure.com/s250246/Figaf/_boards/board/t/Figaf%20Team/Issues?workitem=3",
        "Title": "Ticket initiated in landscape overview from trial (DEV) to trial
(QA)",
        "Description": "Description - Test transport of 2 iflows\n\nTracked Objects:\n-
GetData (1.0.1)\n- TestIflow (1.0.8)",
        "Status": "To Do"
    },
    {
        "ID": "2",
        "URL":
"https://dev.azure.com/s250246/Figaf/_boards/board/t/Figaf%20Team/Issues?workitem=2",
        "Title": "Transport 2",
        "Description": "Second TP",
        "Status": "Doing"
    },
    {
        "ID": "1",
        "URL": "
https://dev.azure.com/s250246/Figaf/_boards/board/t/Figaf%20Team/Issues?workitem=1",
        "Title": "Transport 1",
        "Description": "First transport\n\nTest *transport* _ever_\n\n* Dummy",
        "Status": "Doing"
    }
]
```

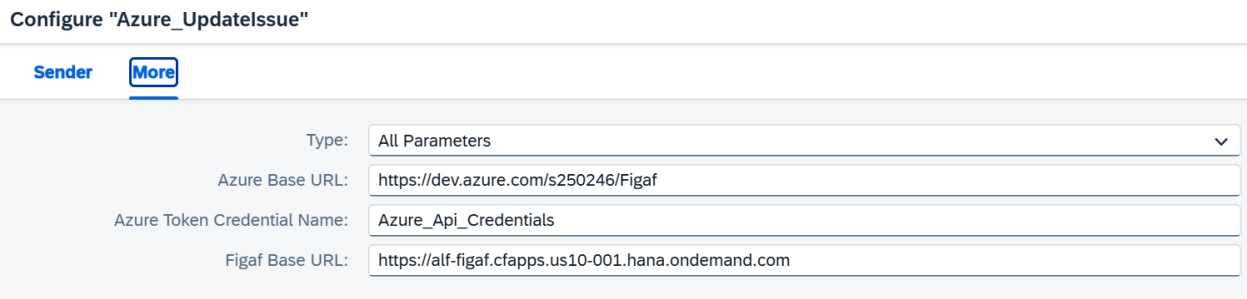
3.2. Azure_UpdateIssue

This iFlow is triggered by webhooks from Figaf whenever tickets or transports undergo changes. Based on a central JSON configuration, it dynamically performs actions in Azure, such as transitioning issue statuses or adding comments.

3.2.1. Configuration (from iFlow's "Configure" tab):



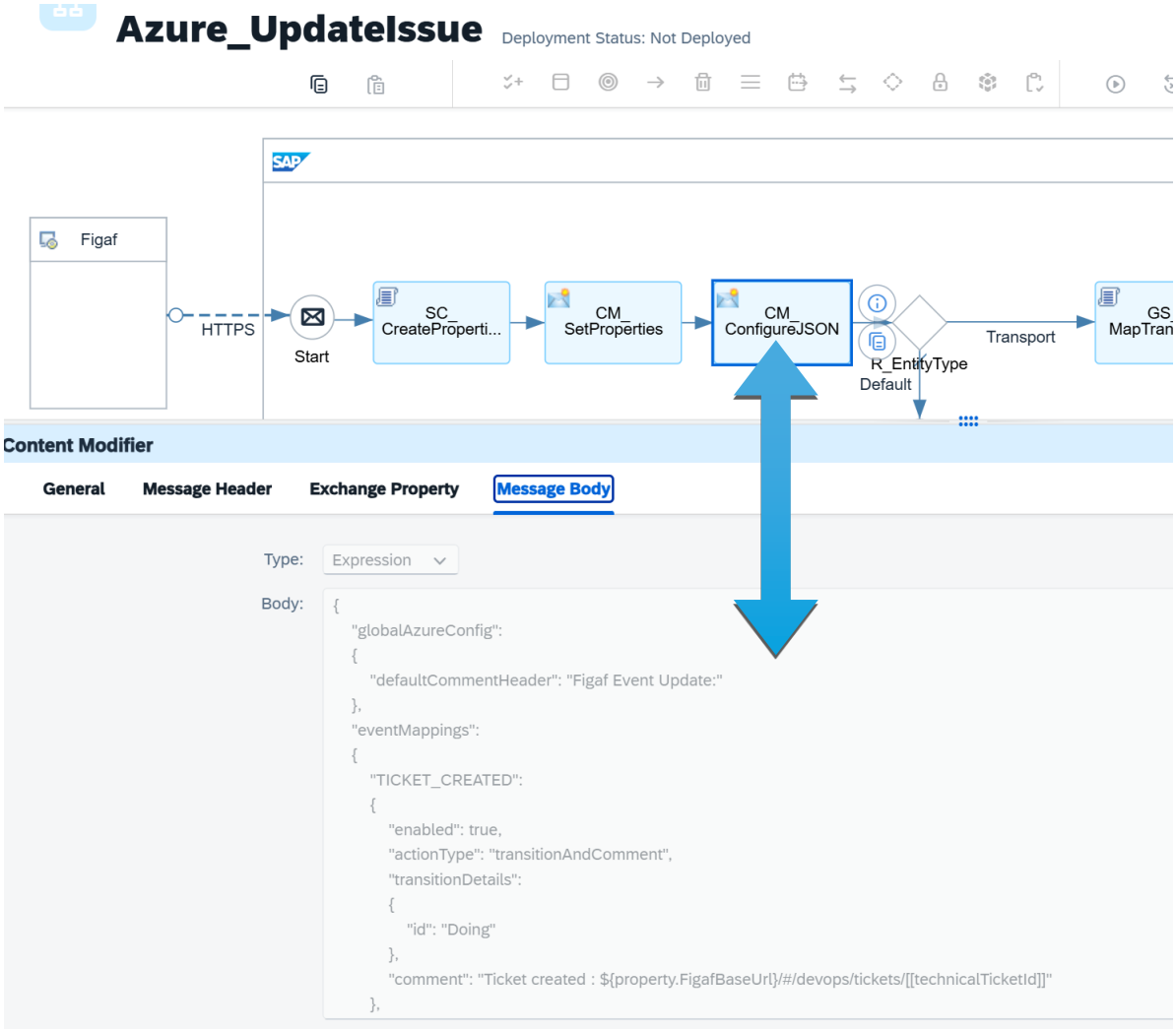
- **Address:** This is the path appended to your Integration Suite tenant URL to call this iFlow.
- **User Role:** ESBMessaging.send in this case (can set and use others if configured and needed).



- **Azure Base URL:** This is the path to your Azure DevOps Project
- **Azure Token Credential Name:** Credentials used to access azure API (email and API key)
- **Figaf Base URL:** Path to your Figaf instance

3.2.2. Options

The primary options for this iFlow are managed through a JSON structure defined within the *CM_ConfigurationJSON* Content Modifier step, specifically on its *Message Body* tab. This JSON dictates how different Figaf events are mapped to Azure actions.



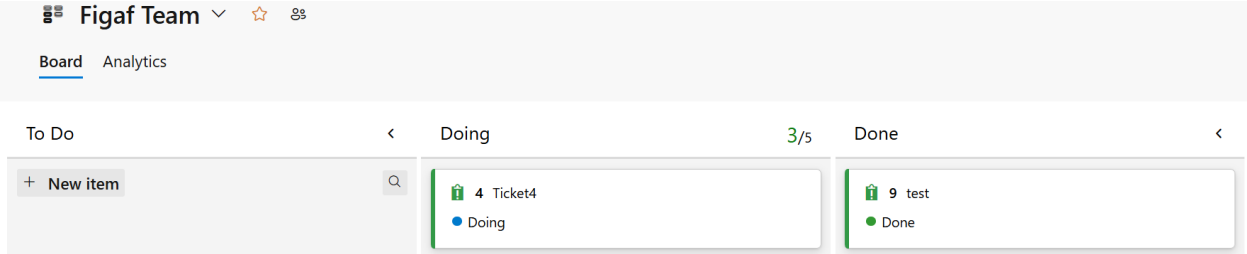
Key Configuration Sections within the JSON:

- **globalAzureConfig**
  - defaultCommentHeader: An optional prefix automatically added to comments posted by this integration (e.g., "Figaf Event Update:").

```
"globalAzureConfig": {
    "defaultCommentHeader": "Figaf Event Update:"
}
```

eventMappings: This is the core section where you define actions for each Figaf eventType. For each event (e.g., TICKET_CREATED, TRANSPORT_APPROVED):

- enabled: (boolean) Set to true to process this event type, false to ignore it.
- actionType: Defines the primary Azure action. Only allowed values:
  - commentOnly: Only adds a comment to the related Azure issue.
  - transitionAndComment: Performs a transition and also adds a comment.
- transitionDetails: (Object, used if actionType involves a transition)
  - id: The name of the Azure workflow state to transition to. Below is a configuration with 3 states: To Do, Doing and Done.



- comment: (String) The template for the comment to be added to the Azure issue. Placeholders in the format [[placeholderName]] will be replaced with values from the Figaf webhook payload.
  - Example: "Transport [[technicalTransportId]] for Figaf ticket [[webhookTicketDtoList[0].technicalTicketId]] is now [[status]]."

This is an example of configuration for the `TRANSPORT_APPROVED` Figaf event.

```
"eventMappings": {
    "TRANSPORT_APPROVED": {
        "enabled": true,
        "actionType": " transitionAndComment",
        "transitionDetails": { "id": "Doing" },
        "comment": "Transport [[technicalTransportId]] for Figaf ticket [[webhook-
TicketDtoList[0].technicalTicketId]] on landscape [[landscape.name]] is now [[sta-
tus]]."
    }
}
```

Explanation:

`"enabled":` **`true`**

The TRANSPORT_APPROVED event will be handled.

`"actionType": " transitionAndComment",`

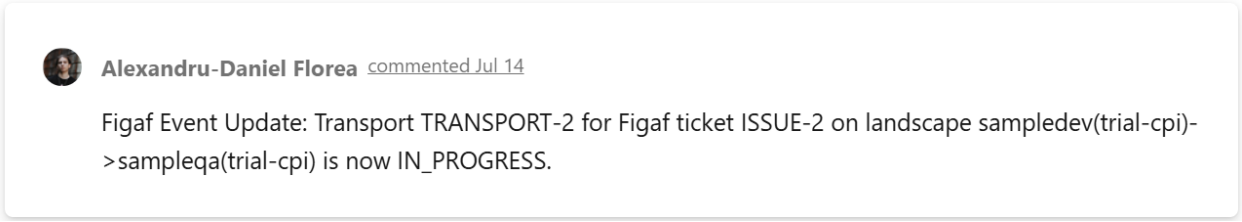
The event will produce a transition and a comment for the corresponding Azure issues.

`"transitionDetails": { "id": "Doing" },`

The new status of the ticket will be that of ID "Doing".

```
"comment": "Transport [[technicalTransportId]] for Figaf ticket [[webhook-TicketDtoList[0].technicalTicketId]] on landscape [[landscape.name]] is now [[status]]."
```

This configuration will generate a comment looking like this:



3.2.3. Using the IFlow

- **Trigger:** HTTPS call from Figaf Webhook.
- **Input:** JSON - This iFlow expects to be called by Figaf webhooks and receive a JSON file.
- **Output:** JSON - Calls to Azure API containing data about transitions and/or comments.

A list of suggested placeholders for comments:

- [[technicalTransportId]]
- [[status]]
- [[landscape.name]]
- [[webhookTicketDtoList[0].externalTicketId]] (for the Azure key if present in the Figaf payload)
- [[webhookTicketDtoList[0].technicalTicketId]] (for the Figaf ticket ID)
- [[figafEntityId]]
- [[eventType]]

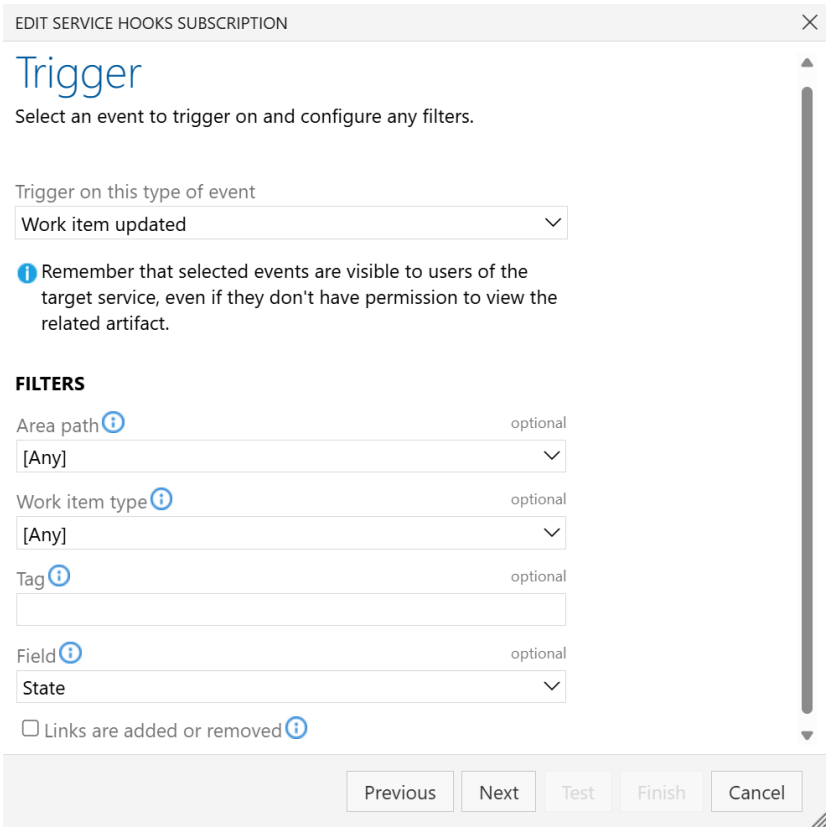
3.3. Azure_FigafUpdateTransport

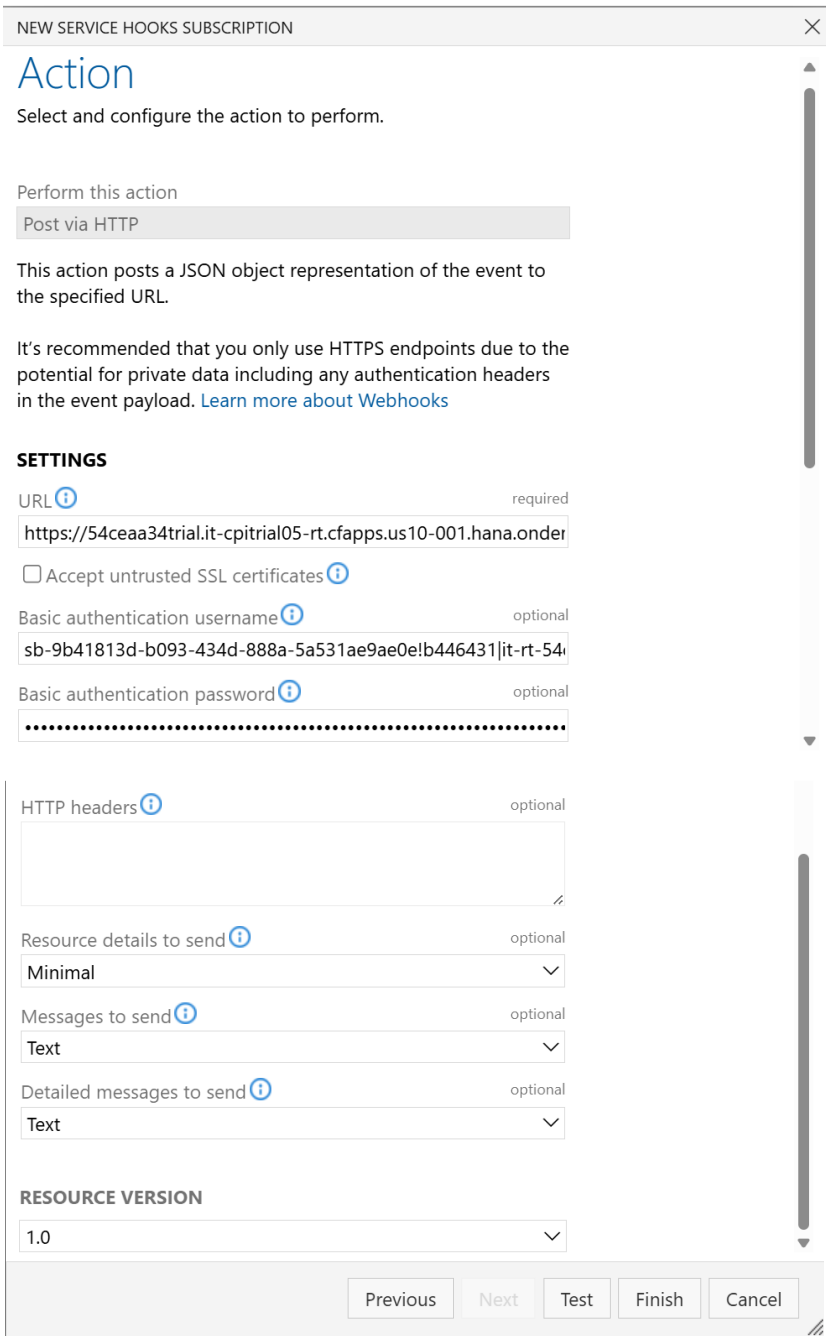
This IFlow provides a way to automatically import a transport in Figaf via a Azure webhook call (for example, moving an issue to “resolved” status).

3.3.1. Setup

3.3.1.1. Create Azure Webhook

Go to Project settings -> General ->Service Hooks -> Create Subscription -> Web Hooks. Configure the webhook like below:





For **URL**, deploy the Azure_FigafUpdateTransport iflow and insert its endpoint.

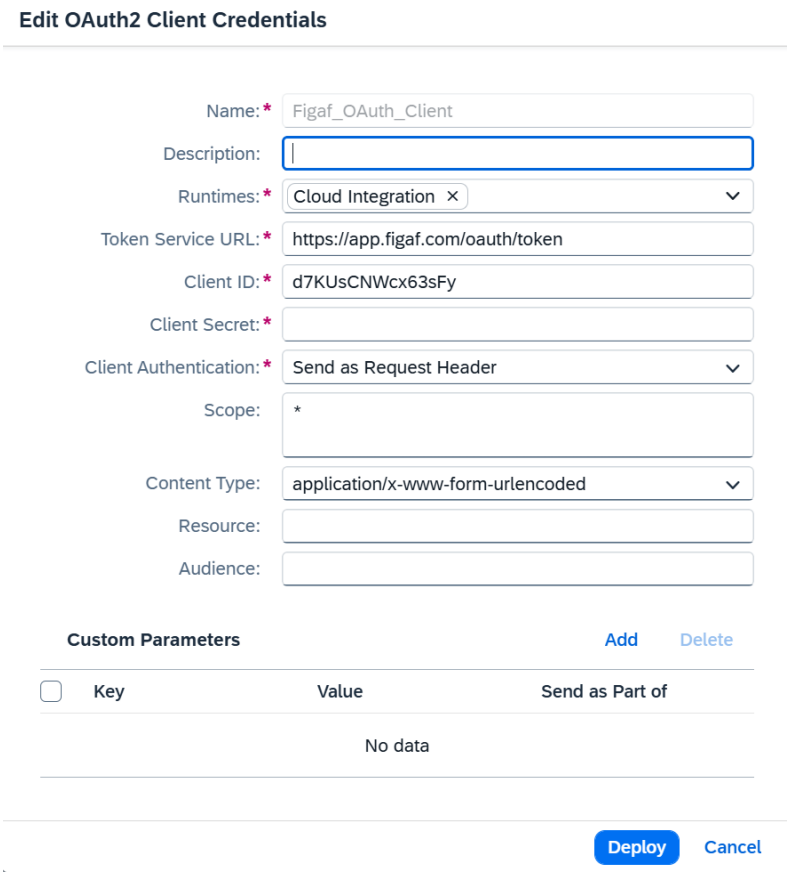
For **username** and **password** use a set of credentials with ESBMessaging.send role.
This can be done in BTP Cockpit by creating a Process Integration Runtime instance, assigning it the ESBMessaging.send role, and generating a service key for it.

3.3.1.1. Create and Add the Figaf Oauth Client credentials

To create the Figaf OAuth Client, in Figaf go to Configuration -> OAuth Clients -> Add OAuth Client. Select the following scopes:

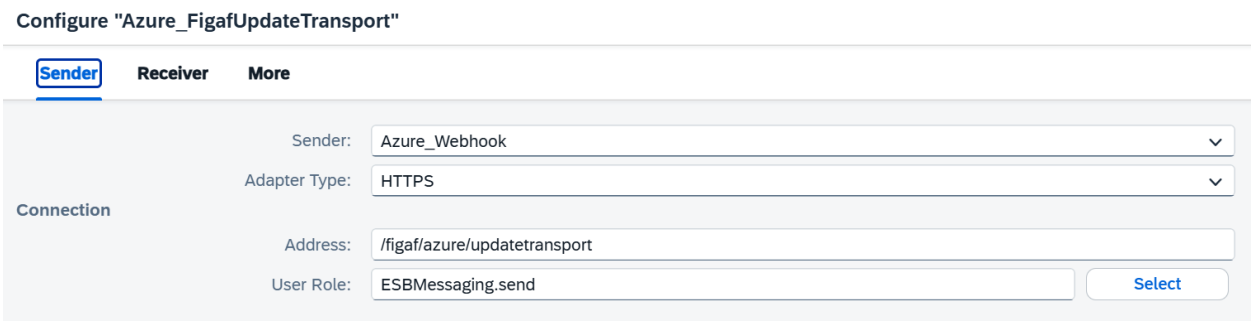
ticket:read, ticket:run, ticket:import, ticket:resolve, transport:read.

Then add the credentials in Integration Suite like below. Token Service Url is your Figaf url followed by /oauth/token

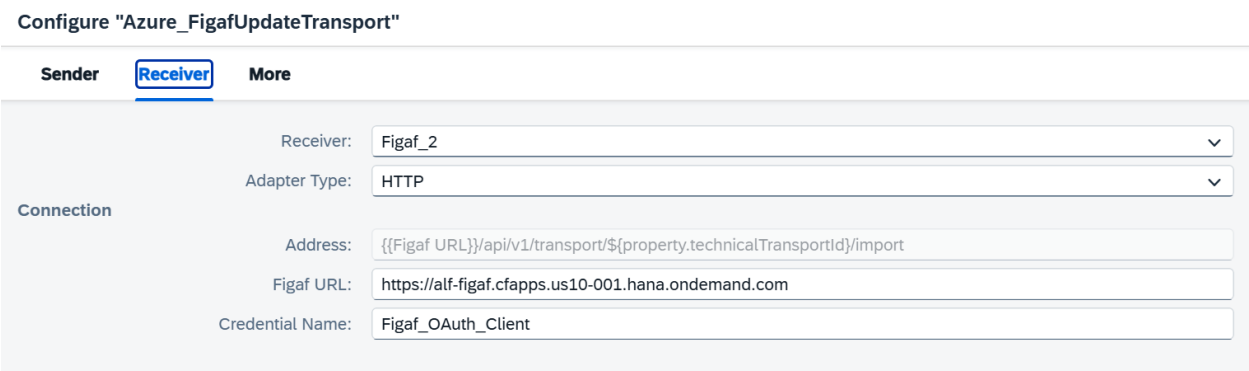


3.3.2. Configuration:

Sender Details:



- **Address:** This is the path appended to your Integration Suite tenant URL to call this iFlow.
- **User Role:** ESBMessaging.send in this case (can set and use others if configured and needed).



- **Figaf URL**: The base Figaf URL. Will be used to build the address to call the API.
- **Credential Name**: The alias for storing the Figaf OAuth Client Credentials.

4. Deployment and General Setup

1. **Deploy Credentials:**
    - Ensure the User Credential artifact containing a valid Azure API token is deployed on your Cloud Integration tenant and that the iFlows configurations reflect the same alias.
    - Ensure the User Credential containing Figaf OAuth Client certificate is deployed.
2. **Configure Azure_UpdateIssue iFlow:**
    - Access the deployed iFlow.
    - Modify the *CM_ConfigurationJSON* Content Modifier's Message Body with your desired eventMappings logic (States names, comment templates, etc.) as detailed in section 3.2.
3. **Configure Azure_SearchIssues:**
    - Access the deployed iFlow and go to the "Configure" view.
    - Update the Receiver Address to your Azure instance URL.
4. **Configure Azure_FigafUpdateTransport:**
    - Follow the Setup Steps from sections 3.3.1.
    - Configure the Figaf URL and the Credential Name.
5. **Deploy Integration Package:** Deploy this integration package ("Figaf and Azure Integration Suite") to your tenant.
6. **Configure Figaf Webhooks:**
    - In Figaf, set up a new integration and then the webhooks to point to the endpoint URL of the **Azure_FigafUpdateTransport** Iflow.