Documentation

Automated Task Synchronization between SAP Cloud ALM and Jira with SAP Integration Suite

SAP Integration Package

Version 1.0

12th December 2025 | Document Version 1.0
RealCore Group GmbH

Contents

Document History

Version	Date	Comments	Affected Pages
1.0	12.12.2025	Initial	All

1. Overview & Introduction

This document describes the SAP Cloud Integration package “SAP Cloud ALM and Jira Task Synchronization”. The package enables bi-directional synchronization of task information between SAP Cloud ALM and Atlassian Jira Cloud. It provides predefined integration flows that exchange task data via REST APIs in both directions and keep the lifecycle of tasks in both systems aligned.

The integration focuses on the functional synchronization of work items such as project tasks and subtasks. It covers the creation, update and deletion or archiving as well as assignment, priority and status changes. The integration is designed so that neither system acts as a permanently leading system. Whenever a task is created in one system, a unique identifier of the corresponding task is stored in both SAP Cloud ALM and Jira. This ensures that subsequent updates or deletions from either side can be clearly assigned to the same business object and automatically trigger the corresponding change in the other system.

To allow proper synchronization of responsible persons for the tasks, the package also contains a dedicated integration flow to preload a value mapping of Jira users. This value mapping is used to translate Jira user account identifiers to SAP Cloud ALM user email addresses and vice versa.

2. Solution Overview

The “SAP Cloud ALM and Jira Task Synchronization” package consists of three main integration flows. Each flow covers a dedicated part of the overall scenario and can be deployed and monitored individually.

The following flows are included:

- Preload Value Mapping for Jira Users – loads Jira users and maintains a value mapping for user assignment.
- Synchronize Tasks from Jira to SAP Cloud ALM – receives Jira events and creates or updates tasks in SAP Cloud ALM accordingly.
- Synchronize Tasks from SAP Cloud ALM to Jira – receives SAP Cloud ALM task events and creates, updates or archives Jira issues.

The package is intended to be operated in a productive landscape where SAP Cloud ALM is used as the central platform for implementation and operations processes, while Jira continues to be used for development and issue tracking. The flows can be adapted to specific customer requirements through configuration of parameters, user mappings and filtering logic.

3. Preparations and what you need

This chapter handles the necessary preliminary steps users need to take care of before using the integration flows.

3.1 Preparations in SAP Cloud ALM

SAP Cloud ALM must be prepared so that it can send outbound task events to Cloud Integration. In this step we will work trough External Management and Project Setup of SAP Cloud ALM.

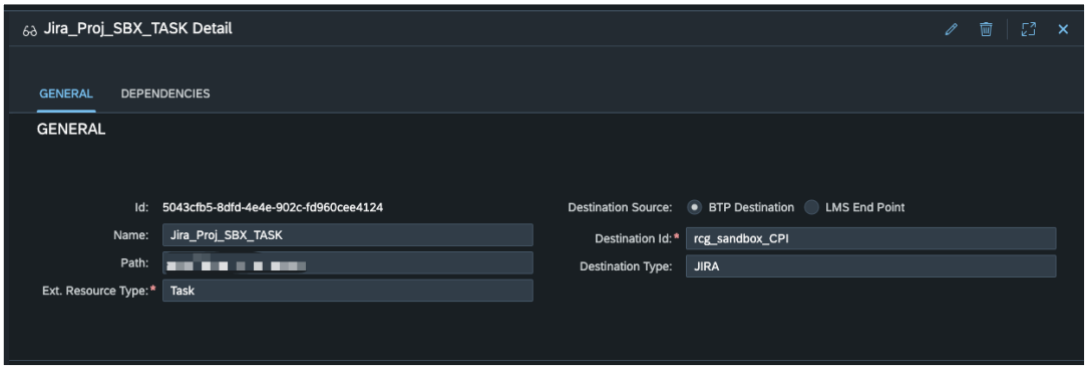
3.1.1 External API Management

External API Management in SAP Cloud ALM is responsible for sending event notifications from SAP Cloud ALM to external consumers such as Cloud Integration.

The configuration in External API Management consists of three main steps, which should be executed in the following order:

Webhook

In the **Webhook** step, the destination that is created in SAP BTP in chapter 3.2.1 is referenced. In addition, a path is maintained that points to the HTTP sender adapter of the respective Integration Flow.

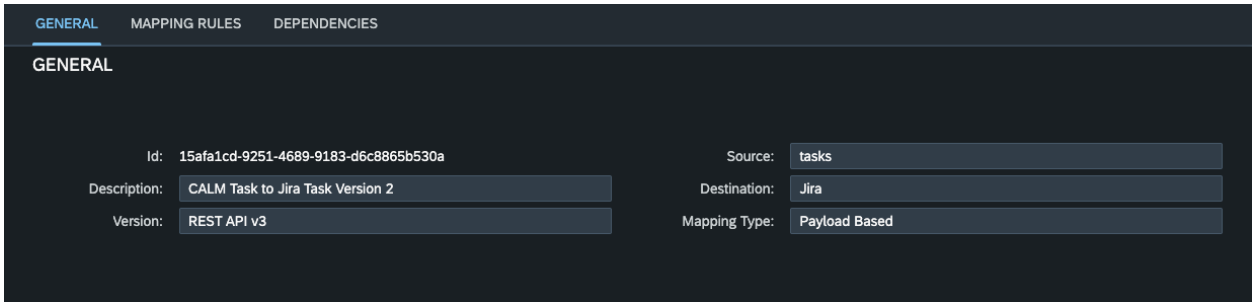


- **Destination ID**: Reference to the destination defined in SAP BTP.
- **Path**: Individual endpoint of the iFlow, for example /http/testendpoint.

This configuration ensures that all relevant events from SAP Cloud ALM are forwarded directly to the correct iFlow in SAP Integration Suite.

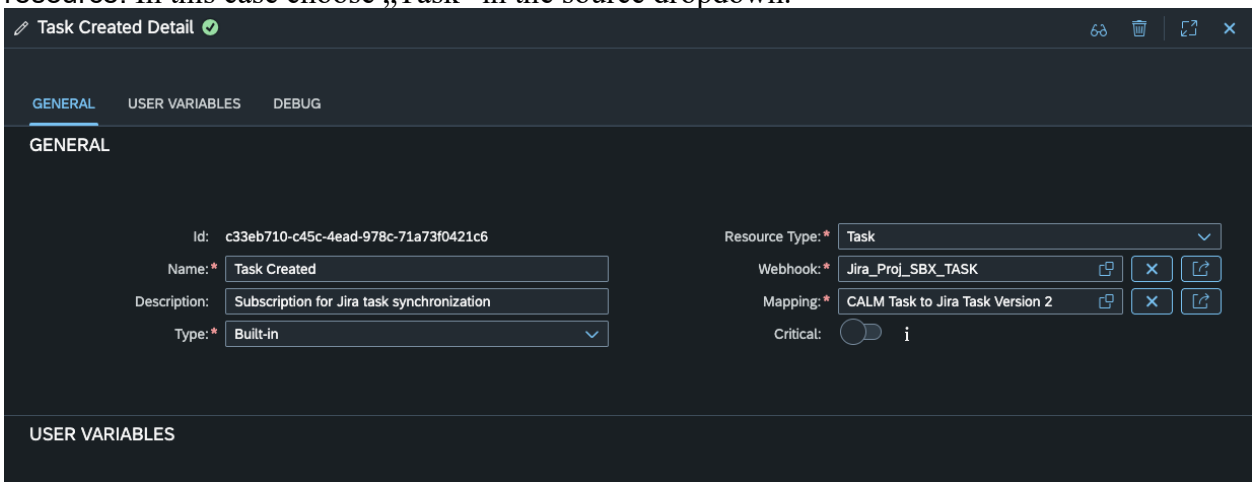
Mappings

In the **Mappings** step, a payload-based mapping is used to control how the event payload from SAP Cloud ALM is transformed before it is sent to SAP Integration Suite.
The provided script from the document **"Payload Based Mapping for SAP Cloud ALM"** section in the Integration Package has to be copied into the mapping editor and saved as a **Payload-Based Mapping**, for example under the name CALM_Jira_Task_Payload_Mapping.
In the general settings following configurations can be choosed:



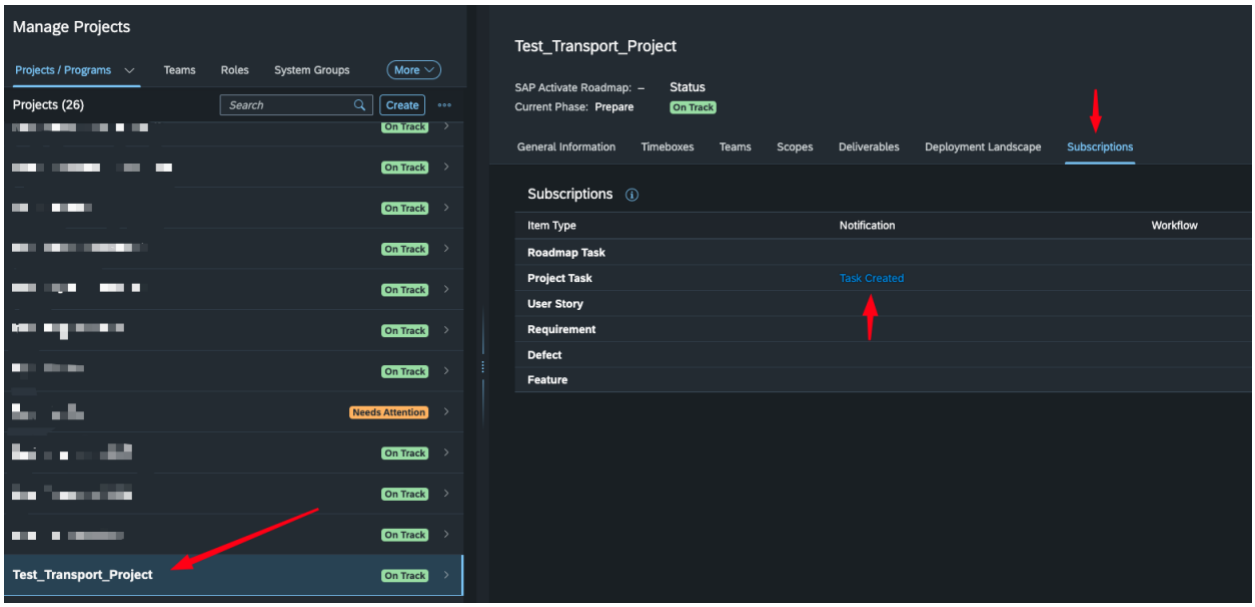
Subscription

Finally, a **Subscription** is created. A subscription links the previously configured Webhook with the Mapping and thereby activates the delivery of the desired events for the respective resource. In this case choose „Task" in the source dropdown.



3.1.2 Project and Setup

In the final step, open the relevant project in SAP Cloud ALM and assign the previously created subscription to the resource type *Tas*k.



In this view, select the required project and copy the projectId from the browser URL. You must keep this ID for the later configuration of the integration.

3.2 Preparations in SAP Business Technology Platform

This section describes the fundamental prerequisites in **SAP Business Technology Platform (BTP)** that must be in place to establish the integration between SAP Cloud ALM and Jira via SAP Integration Suite.

3.2.1 BTP Destination

A destination in SAP BTP is required so that SAP Cloud ALM communicates with Jira through SAP Cloud Integration. This destination will be used by the external API management from SAP Cloud ALM.

Key requirements:

- The destination **must be created in the same BTP subaccount** in which SAP Cloud ALM is consumed as a service.
- The **URL** of the destination must point to the **base HTTP URL of the Cloud Integration tenant** (not yet to a specific Integration Flow endpoint).
- The **Integration Flow-specific path** (for example /http/CloudALM2Jira) is added later in SAP Cloud ALM as part of the Webhook configuration.
- Authentication for the destination **must be based on Basic Authentication or a Service Key** of the Cloud Integration (see next chapter) to ensure a secure and stable connection.

3.2.2 BTP Service Keys

To enable secure communication between the systems, appropriate **Service Keys** must be created in SAP BTP.

Cloud Integration Service Key

For the destination authentication, you must create a **Service Key** for the SAP Integration Suite instance with **read and write permissions**. This service key must be created in the BTP subaccount where the target Cloud Integration tenant is running. This key is then referenced by the destination used by SAP Cloud ALM when calling the Webhook endpoint in SAP Integration Suite.

SAP Cloud ALM Service Key

To allow Cloud Integration to authenticate against SAP Cloud ALM, you must also create a **Service Key in the SAP Cloud ALM tenant**.
This key must provide sufficient permissions to:

- Read tasks and projects
- Create tasks
- Update tasks
- Delete or mark tasks as obsolete

3.3 Preparations in Atlassian Jira Cloud

To integrate Jira with SAP Cloud ALM, both inbound and outbound integration prerequisites in Jira must be fulfilled. The following sections describe the steps to create the Jira API token and to configure the Jira Automation rules in detail.

3.3.1 Jira API Key

For Basic Authentication from Cloud Integration to Jira, a **Jira API token** must be created.
Proceed as follows:

1. Sign in to Atlassian at https://id.atlassian.com/manage-profile/security/api-tokens.
2. Create a new API token (for example via **Create API token** or **Create scoped API token**).
3. Assign a meaningful name that clearly describes the purpose of the token, for example CPI_CloudALM_Integration.
4. Define an expiry date for the token.
   The validity period must be chosen according to your security policy (typically between 1 and 365 days).
5. Select the application the token shall access (Jira Cloud) and assign the required scopes/permissions, for example: write:jira-work, read, delete or give all permissions
6. Confirm the creation of the token.
7. Copy the generated token to the clipboard and store it **securely**.
   The token must then be maintained in SAP Cloud Integration as a **User Credential** under security materials.

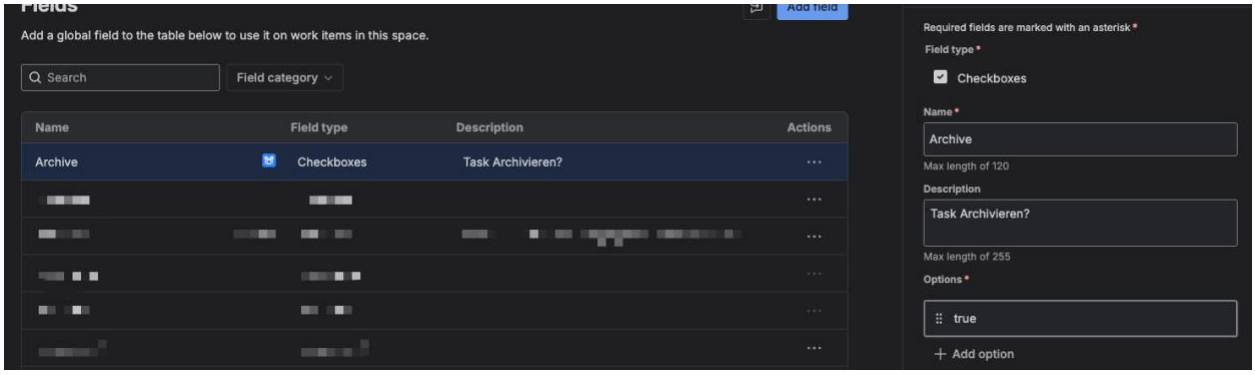
   This API token is used by all integration flows that call Jira REST APIs for creating, updating or archiving issues.

3.3.2 Jira Automation

In the **Project settings** of each Jira project that must be integrated with SAP Cloud ALM, you have to navigate to **Automation** and configure the required rules.
For the described integration scenario, **four automation rules** are required.
Jira Automation webhooks support **Basic Authentication only**. Therefore, the webhook configuration must use either:

- the technical SAP and Jira user account or
- a dedicated **Service Key with SAP Cloud Integration permission**

Important:
Additionally create a new custom field for issue items in your project space. This item should be named “Archive” from filed type checkboxes with only one value option “true”.

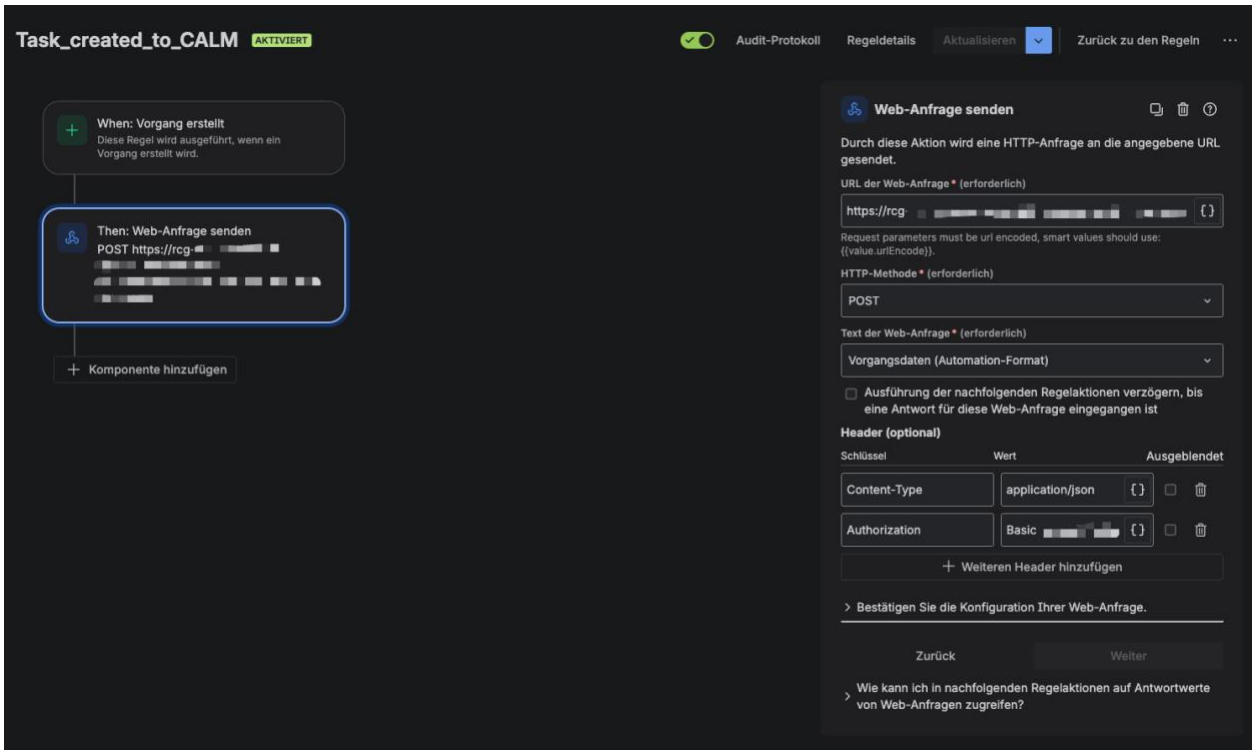


Navigate to project setup and open „Automation“. Create the following Automations.

3.3.2.1 Create – Automation

The **Create** automation has the purpose of triggering the synchronization between Jira and SAP Cloud ALM every time an issue is created in the Jira project.
The rule **must** be configured as follows:

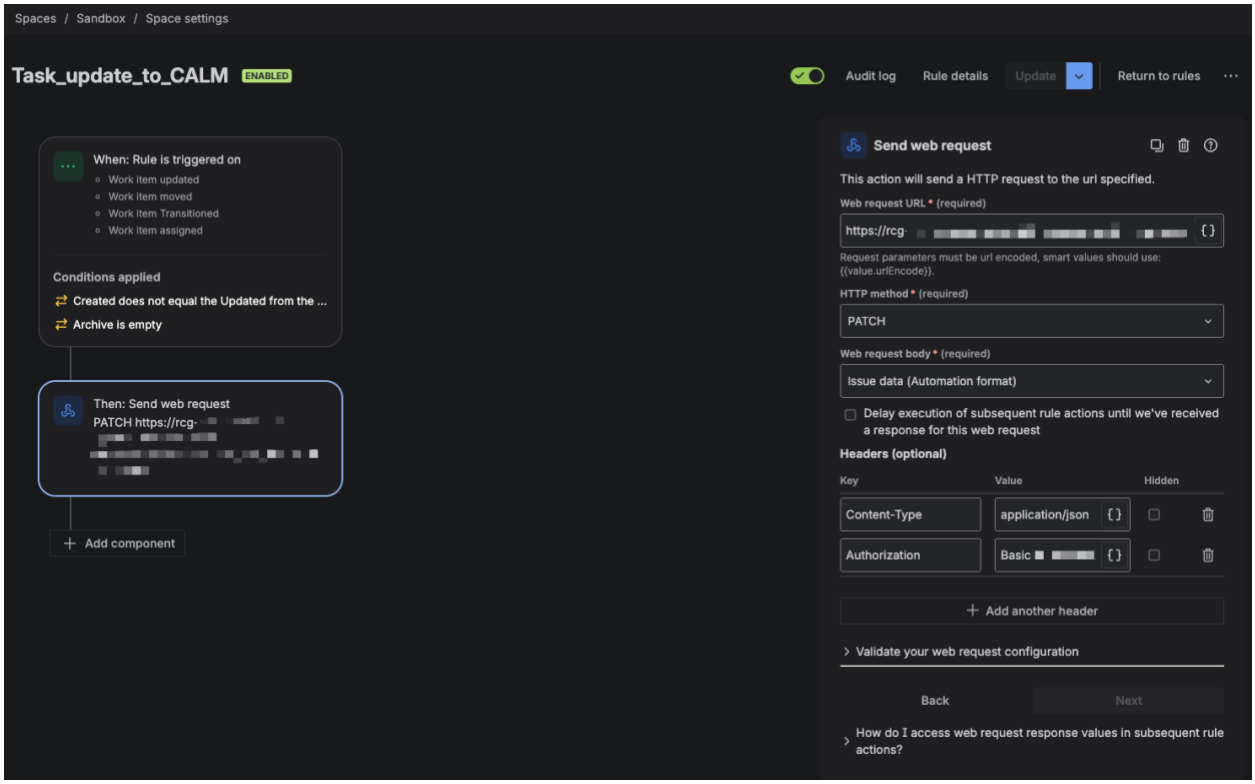
1. Trigger: When Work Item created
2. Send web Request
    a. Web request URL: your Integration Flow HTTP endpoint
    b. HTTP Method: POST
    c. Web request body: Issue data
    d. Headers: Content-Type : application/json
    Authorization : Basic {Base64Encoded **User:Password**}



3.3.2.2 Update – Automation

The Update automation triggers the synchronization between Jira and SAP Cloud ALM whenever an existing issue is changed in the Jira project.
The rule must be configured as follows:

1. Trigger: When Work Item updated, moved, transitioned, assigned
2. Add Conditions: Created does not equal Updated and Field:Archive is empty
3. Send web Request
    a. Web request URL: your Integration Flow HTTP endpoint
    b. HTTP Method: PATCH
    c. Web request body: Issue data
    Headers: Content-Type : application/json
    Authorization : Basic {Base64Encoded **User:Password**}



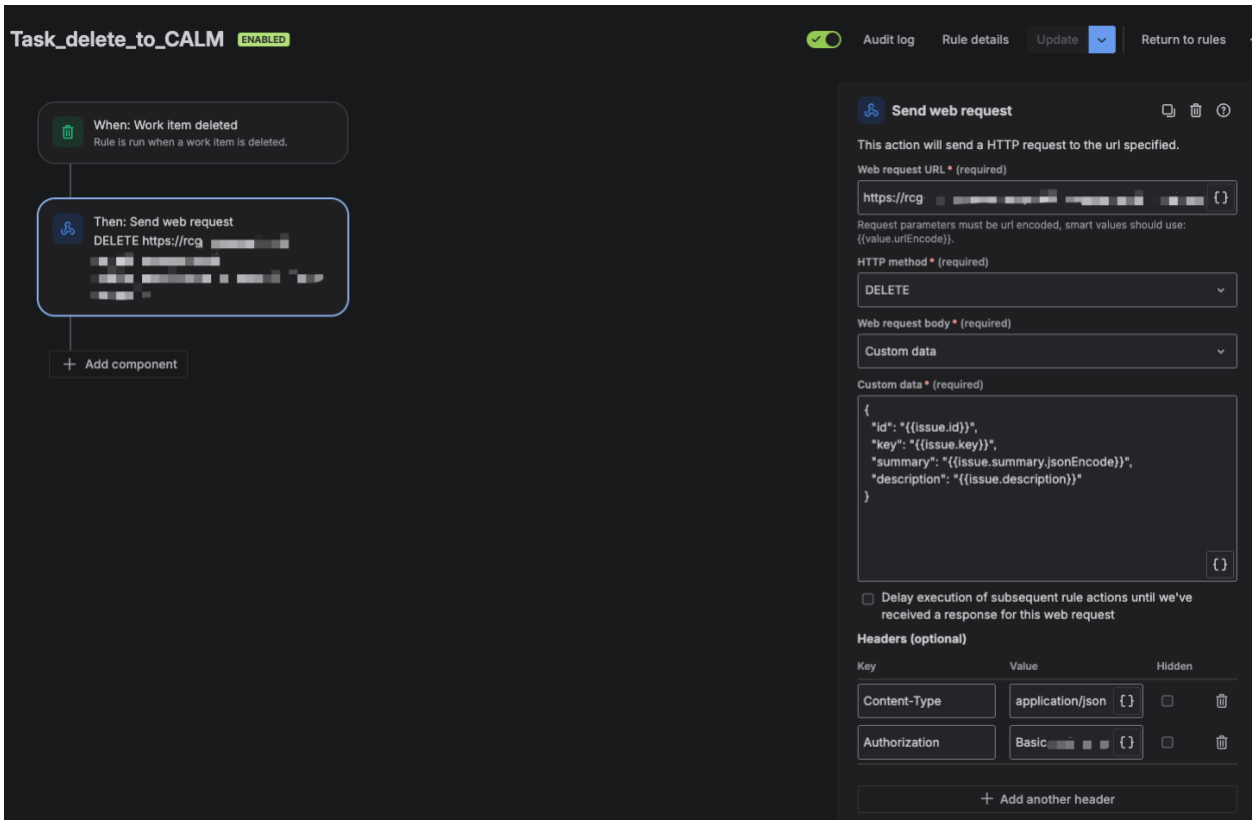
3.3.2.3 Delete – Automation

The **Delete** automation ensures that the synchronization between Jira and SAP Cloud ALM is also triggered when an issue is deleted in the Jira project.
The rule **must** be configured as follows:

1. Trigger: When Work Item deleted
2. Send web Request
    a. Web request URL: your Integration Flow HTTP endpoint
    b. HTTP Method: Delete
    c. Web request body: Custom Data:

```
{
 "id": "{{issue.id}}",
 "key": "{{issue.key}}",
 "summary": "{{issue.summary.jsonEncode}}",
 "description": "{{issue.description}}"
}
```

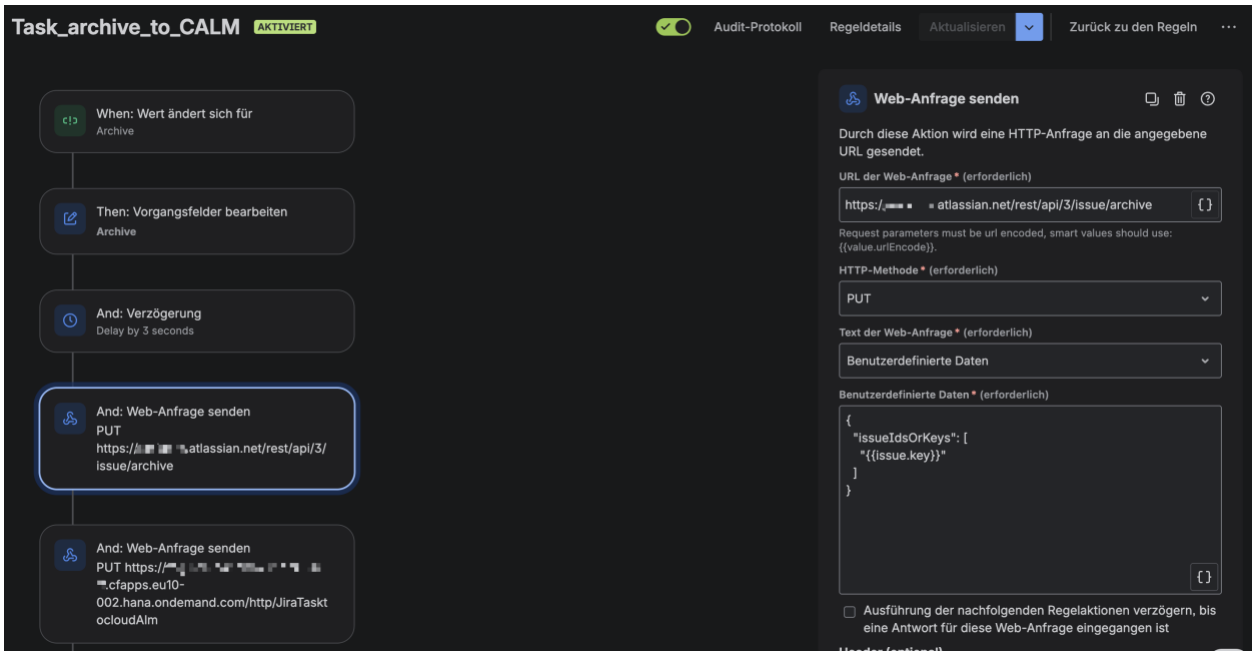
    d. Headers: Content-Type : application/json
    Authorization : Basic {Base64Encoded **User:Password**}

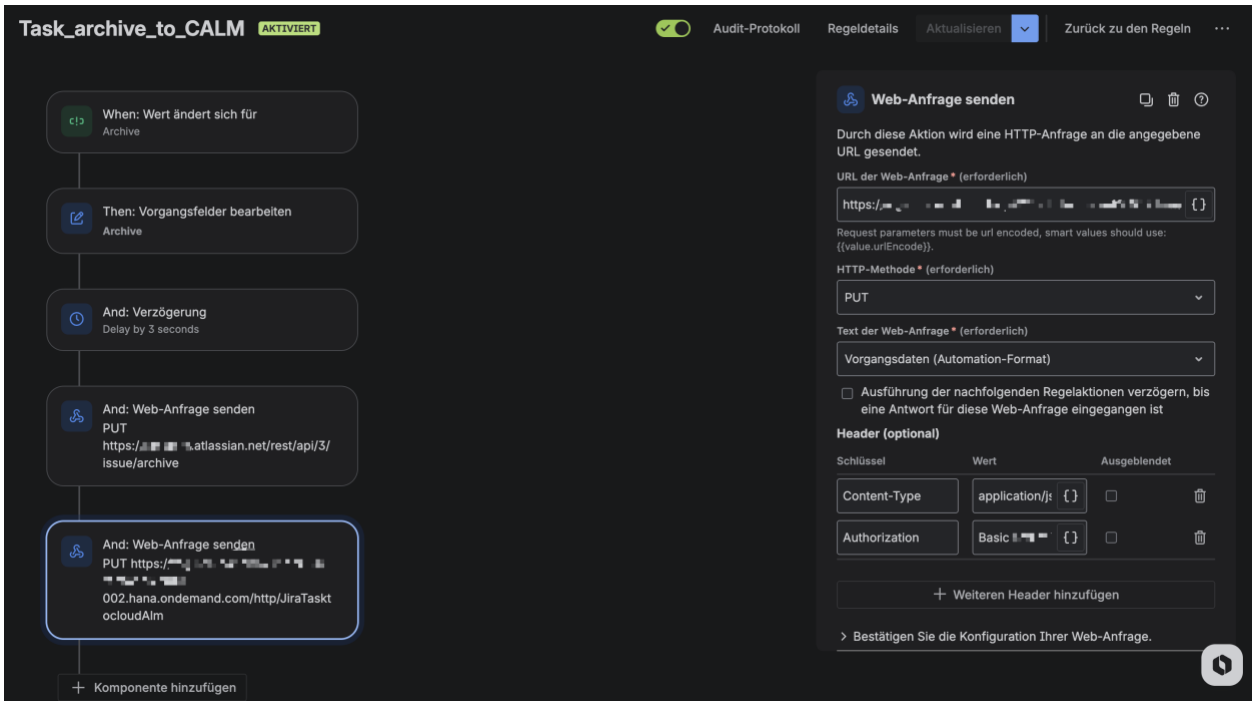


3.3.2.4 Archive – Automation

The Archive automation ensures that, whenever an issue is flagged for archiving in Jira, the issue is archived in Jira and the corresponding task in SAP Cloud ALM is updated accordingly. The rule must be configured as follows:

1. **Trigger:** Field value changed for the custom field Archive, with change type set to **Value added**.
2. **Action:** Edit Work item - field „Archive“ set empty
3. **Action:** Add a short **Delay** of 3 seconds
4. **Send web request**
   a. Web request URL: your Atlassian Archive API
      - https://{your-domain}.atlassian.net/rest/api/3/issue/archive
   b. HTTP Method: Put
   c. Web request body: Custom Data
      ```
      {
        "issueIdsOrKeys": [
          "{{issue.key}}"
        ]
      }
      ```
   d. Headers: Content-Type : application/json
      Authorization : Basic {Base64Encoded **User:Password**}
5. **Send web request**
   a. Web request URL: your Integration Flow HTTP endpoint
   b. HTTP Method: PUT
   c. Web request body: Issue data
   d. Headers: Content-Type : application/json
      Authorization : Basic {Base64Encoded **User:Password}**





4. IFlows and Configuration in Cloud Integration

This packages comes with three integration flows:

1. Main Flow "*Synchronize Tasks from Jira to SAP Cloud ALM*"
2. Main Flow "*Synchronize Tasks from SAP Cloud ALM to Jira*"
3. Optional Flow "*Preload Value Mapping for Jira Users*"

This documentation primarily focuses on the Main Flows as the third one is an optional addition which can be used to enable the task assignment for the main flows (more details for the value mapping flow can be found in chapter 4.5).

4.1 Configuration Parameters – Main Flows

To fully use the integration flow the following parameters needs to be configured first before Deployment:

Synchronize Tasks from Jira to SAP Cloud ALM

Category	Name	Default	Mandatory parameter (X)	Description
SAP Cloud ALMConnection	SAP Cloud ALM Base URL		X	Enter here the base SAP Cloud ALM URL of your tenant. Example: **example-calm.eu10.alm.cloud.sap**
	SAP Cloud ALM Credentials		x	Name of the Security Material created for SAP Cloud ALM Oauth2 Authentification. See chapter 3.2.2 - SAP Cloud ALM Service Key

	SAP Cloud ALM Project ID		X	Enter the SAP Cloud ALM Project ID of the project for which you want to enable task synchronization. The Project ID is a 36-character GUID in the format
Jira Connection	Archive Field Jira Id		X	Enter here the archive field ID, which was created in chapter 3.3.2
	Jira Base URL		X	Enter here you base URL of your Atlassian Jira Cloud Account Example: **example.atlassian.net**
	Jira_Basic_Credentials		X	Enter the name of the Security Material that contains the Basic Authentication credentials for Jira. The username must be the e-mail address of the user who created the API token, and the password must be the API token itself. API Token created in chapter 3.3.1
Status Mapping (optional)	Status Open	none		Enter here the ***status ID*** of the Jira "to do" status, which will be mapped to CIPTKOPEN Status in SAP Cloud ALM Note: You can find the five digit status ID's in the task payload under: "status": { "untranslatedNameValue": null, "self": "https://atlassian.net/rest/api/2/status/11308", "description": "", "iconUrl": "https://atlassian.net/", "name": "Done", "untranslatedName": null, "id": 11308, "statusCategory": …. or by calling this GET API: {your.jiradomain}/rest/api/3/status you will get a list, of all possible statuses in your Jira.

	Status In Progress	none		Enter here the ***status ID*** of the Jira "In Progress" status, which will be mapped to CIPTKINP Status in SAP Cloud ALM
	Status Blocked	none		Enter here the ***status ID*** of the Jira "Blocked" status, which will be mapped to CIPTKBLK Status in SAP Cloud ALM
	Status Done	none		Enter here the ***status ID*** of the Jira "Done" status, which will be mapped to CIPTKCLOSE Status in SAP Cloud ALM
	Status Not Relevant	none		Enter here the ***status ID*** of the Jira "Not Relevant" status, which will be mapped to CIPTKNO Status in SAP Cloud ALM

Note: It is important to fill the five digit status id's of Jira to enable the status synchronization. Otherwise a fallback in the Script will synchronize the statuses with following rules:

```
String mapJiraStatusToSapCloudAlm(String name) {
   if (!name) return null
   def s = name.toLowerCase().trim()
   if (s in ["to do","open","neu","new","backlog"])              return "CIPTKOPEN"
   if (s in ["in progress","in arbeit","doing","in process"])    return "CIPTKINP"
   if (s in ["blocked","on hold","wartend","waiting"])           return "CIPTKBLK"
   if (s in ["not relevant","won't do","won't do","wont do","irrelevant"]) return "CIPTKNO"
   if (s in ["done","closed","resolved","fertig","erledigt"])    return "CIPTKCLOSE"
   return "CIPTKOPEN"
}
```

Synchronize Tasks from SAP Cloud ALM to Jira

Category	Name	Default	Mandatory parameter (X)	Description
SAP Cloud ALMConnection	SAP Cloud ALM URL		X	Enter here the base SAP Cloud ALM URL of your tenant. Example: **example-calm.eu10.alm.cloud.sap**
	SAP Cloud ALM Credentials		x	Name of the Security Material created for SAP Cloud ALM Oauth2 Authentification. See chapter 3.2.2 - SAP Cloud ALM Service Key

Controll	Unassign Users	true		Choose **"true"** or **"false"** to control whether the assignee in Jira is removed when no responsible user is set in SAP Cloud ALM. • If set to **"true"**, the Jira issue will be **unassigned** when the responsible field in SAP Cloud ALM is empty. • If set to **"false"**, the current Jira assignee will **remain unchanged**, even if the responsible field in SAP Cloud ALM is empty.
Jira Connection	Jira Projekt Key		X	Insert the Project Key of the Project that you would like to connect to the ticket synchronization
	Jira URL		X	Enter here the base URL of your Atlassian Jira Cloud Account Example: example.atlassian.net
	Jira_Basic_Credentials		X	Enter the name of the Security Material that contains the Basic Authentication credentials for Jira. The username must be the e-mail address of the user who created the API token, and the password must be the API token itself. API Token created in chapter 3.3.1
	Archive Field Jira Id		X	Enter here the archive field ID, which was created in chapter 3.3.2 **Important:** The value must include the Jira prefix customfield_ in front of the numeric ID. Example: **customfield_10650**
	Archive true option Id		X	Enter here the customfield true option Id, which was created in chapter 3.3.2
	Jira Issue Type		X	Enter the ID of the Jira *Issue Type* in the project that you want to synchronize. Example: **19873**
	Jira Issue Type Sub-Task		X	Enter the ID of the Jira *Sub-Task Issue Type* in the project that you want to synchronize as sub-tasks.

				Example: **19874** (project-specific value)

Status Mapping (optional)	Status Open	none		Enter here the ***transition ID*** of the Jira "to do" transition, which will be equivalent to CIPTKOPEN Status from SAP Cloud ALM
				
				Note: You can find the one or two digit transition ID's by calling this API:
				
				GET {your.jiradomain}/rest/api/3/issue/{IssueKeyOrId}/transitions
				You will get following payload:
				{
				"transitions": [
				{
				"id": "21",
				"name": "To Do",
				"to": {
				"id": "11306",
				"name": "New"
				}
				}
				]
				}
				
				Note: Send the GET request for a task that is currently in the *Open* state. Repeat this for tasks that are currently in the respective target states when configuring the other status mappings.

	Status In Progress	none		Enter here the ***transition ID*** of the Jira "In Progress" transition, which will be equivalent to CIPTKINP Status from SAP Cloud ALM
	Status Blocked	none		Enter here the ***transition ID*** of the Jira "blocked" transition, which will be equivalent to CIPTKBLK Status from SAP Cloud ALM

	Status Closed	none		Enter here the ***transition ID*** of the Jira "done" transition, which will be equivalent to CIPTKCLOSE Status from SAP Cloud ALM
	Status Not Relevant	none		Enter here the ***transition ID*** of the Jira "not relevant" transition, which will be equivalent to CIPTKNO Status from SAP Cloud ALM

4.2 Create Sub-Flow

Both integration flows contain a *Create* sub-flow. This sub-flow is responsible for transforming the incoming payload into the target system format, creating the corresponding task or sub-task, and writing back the external ID to establish the cross-reference between SAP Cloud ALM and Jira. This section briefly describes the behaviour of these create sub-flows.

4.2.1 SAP Cloud ALM to Jira

The *Create* sub-flow in the SAP Cloud ALM to Jira iFlow first checks whether the incoming SAP Cloud ALM task is a standard task or a sub-task and routes the message accordingly. For normal tasks it maps the payload to the Jira issue structure, sends a create request to Jira and receives the Jira key/ID in the response. For sub-tasks it also resolves the Jira parent issue and builds a sub-task payload that is linked to this parent. Finally, a follow-up step sends a small payload back to SAP Cloud ALM so that the direct Jira link and external reference are stored on the original task.

4.2.2 Jira to SAP Cloud ALM

The *Create* sub-flow in the Jira to SAP Cloud ALM iFlow processes create events coming from Jira Automation. The incoming Jira issue is mapped to the SAP Cloud ALM task structure and a router checks whether the issue already contains a reference to an existing SAP Cloud ALM task; if so, the creation branch is skipped. This is necessary because Jira Automation triggers when SAP Cloud ALM is creating a issue via API. For new items, the flow distinguishes between standard issues and sub-tasks: sub-tasks retrieve the parent issue and resolve the corresponding parent task in SAP Cloud ALM, while normal issues are created directly. After the SAP Cloud ALM task has been created, a follow-up step sends an update back to Jira so that the Cloud ALM task ID and deep link are stored in the task description.

4.3 Update Sub-Flow

Both integration flows contain an Update sub-flow that processes change events and keeps SAP Cloud ALM and Jira aligned over the full task lifecycle. The update logic evaluates the current state in both systems, decides whether a normal field update, an archive or an unarchive operation is required, and then calls the appropriate API endpoints.

4.3.1 SAP Cloud ALM to Jira

The Update sub-flow in the SAP Cloud ALM to Jira iFlow starts by mapping the incoming SAP Cloud ALM task event to a Jira update structure. It then reads the current Jira issue, including the *softArchived* information, and uses the status of the SAP Cloud ALM task (obsolete flag) together with the Jira archive state to decide whether the issue must be updated, archived or unarchived. Based on this decision, the flow either calls the Jira archive endpoint, triggers an unarchive call followed by a normal update, or sends a pure update request.

NOTE: Jira will only change the status/transitions, which are allowed within the project. Since in SAP Cloud ALM its possible to transit from any status to any ticket status directly, in Jira it may not be possible.

4.3.2 Jira to SAP Cloud ALM

The Update sub-flow in the Jira to SAP Cloud ALM iFlow processes update events that are sent from Jira Automation. The incoming Jira issue is first mapped to a SAP Cloud ALM task patch structure, after which the flow retrieves the current version of the corresponding task from SAP Cloud ALM. A decision step compares the incoming changes with the existing task (including the obsolete/archived status) and builds the final payload that must be applied to SAP Cloud ALM. The iFlow then sends the update request to the SAP Cloud ALM task API and stores the response.

4.4 Delete Sub-Flow

The Delete sub-flows handle the removal of tasks and ensure that deletions are consistently reflected in both SAP Cloud ALM and Jira. Delete events are always driven by the system in which the action was initiated, while the corresponding sub-flow calls the appropriate API of the opposite system and logs the technical response for audit and troubleshooting purposes.

4.4.1 SAP Cloud ALM to Jira

The flow first maps the event to a simple structure containing the Jira issue identifier and, if required, performs an unarchive call to Jira to ensure that the issue is in a deletable state. It then sends a delete request to the Jira REST API and prepares a compact response payload, which is written to the log for traceability.

NOTE: Deleting parent task without deleting als of the sub-task in SAP Cloud ALM beforehand, will lead to an Error in Jira You

4.4.2 Jira to SAP Cloud ALM

The flow first resolves the corresponding SAP Cloud ALM task ID (UUID) and then calls the SAP Cloud ALM task API to delete or mark the task as obsolete, depending on the integration design.

4.5 Preload Value Mapping for Jira Users

The **“Preload Value Mapping for Jira Users”** integration flow is required to enable task assignment in both systems.

Both main synchronization flows call the Value Mapping API to resolve the user between SAP Cloud ALM and Jira. If this value mapping is not available or not used, the assignee information is skipped and the tasks are created or updated without an assignee in the target system.

4.5.1 Configuration

To fully use the integration flow the following parameters needs to be configured first before Deployment:

Value Mapping for Jira Users

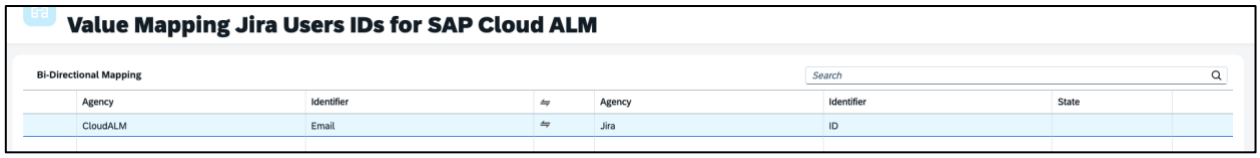
Category	Name	Default	Mandatory parameter (X)	Description

Jira Connection	Project Key		X	Enter the project key of the Jira project from which all users are to be read
	Jira Address		X	Enter here you base URL of your Atlassian Jira Cloud Account Example: **example.atlassian.net**
	Jira Credentials		X	Enter the name of the Security Material that contains the Basic Authentication credentials for Jira. The username must be the e-mail address of the user who created the API token, and the password must be the API token itself. API Token created in chapter 3.3.1
SAP Cloud Integration Connection	SAPCI_Authentification (tab receiver: ”Authentication”)		X	Choose type of authentication to read Cloud Integration objects. Valid values: Basic OAuth2 client credentials
	SAPCI_Credentials (tab receiver: “Credential Name“)		X	Name of the Security Material created for calling cloud integration API. See chapter 3.2.2 – SAP Cloud Integration
	ODATA_Credentials		X	Name of the Security Material created for calling cloud integration ODATA API’s. See chapter 3.2.2 – SAP Cloud Integration
	SAPCI_Hostname		X	Hostname of your SAP Cloud Integration OData API Url Example: **example.cfapps.eu10-002.hana.ondemand.com**

Value Mapping Creation	Value Mapping - Package ID			Integration Package ID fitting the Naming Policies in the Tenant. Only needed for initial Run.
	Value Mapping - Package Name			Integration Package Name fitting the Naming Policies in the Tenant. Only needed for initial Run.
	Value Mapping ID		X	Value Mapping ID fitting the Naming Policies in the Tenant. This ID is used to find the Value Mapping after it has been created. Wrong ID leads to creation of a new Value Mapping in the Tenant.
	Value Mapping Name			Value Mapping Name fitting the Naming Policies in the Tenant.

4.5.2 Value Mapping Visualization

The Value Mapping consists of one transformations:
SAP Cloud ALM to Jira User ID



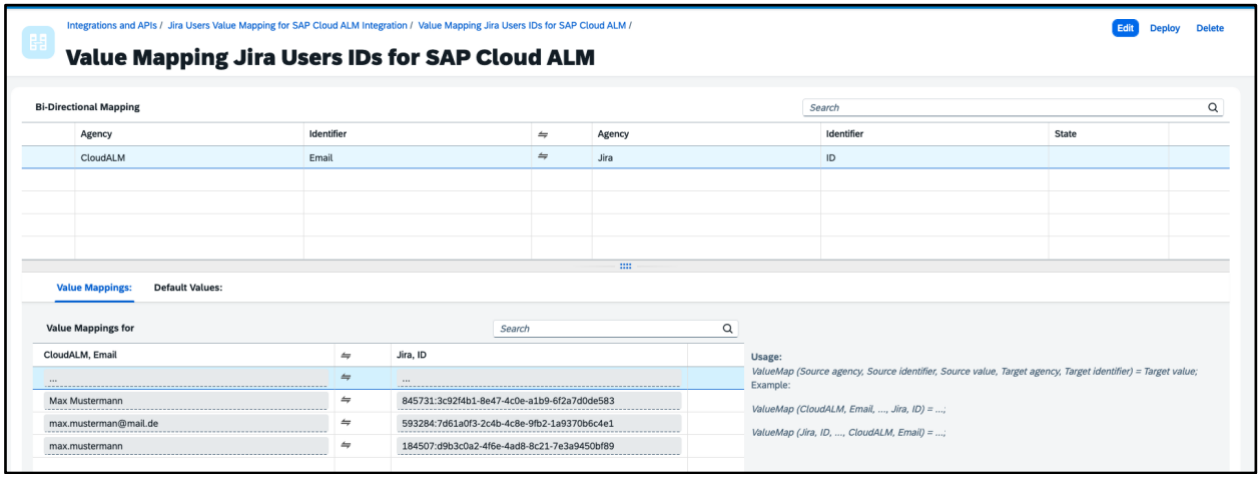
Here an example how this works:
The Jira user IDs are preloaded from Jira into the value mapping. The main objective is to maintain the **correct e-mail address** that belongs to each Jira user ID.
If no e-mail address is available in Jira, the user name is written instead.

As the value mapping maintainer, you **must validate** and, if needed, **correct** the entries in the **Email** colum so that they match the SAP Cloud ALM user e-mail addresses.

In this example:

- The following entry is already correct:
  max.mustermann@mail.de → 593284:7d61a0f3-2c4b-4c8e-9fb2-1a9370b6c4e1

- The following entries must be adjusted to the correct SAP Cloud ALM e-mail address of the user:
  Max Mustermann → 845731:3c92f4b1-8e47-4c0e-a1b9-6f2a7d0de583
  max.mustermann → 184507:d9b3c0a2-4f6e-4ad8-8c21-7e3a9450bf89



Replace these values in the **Email** column with the correct e-mail addresses from the respective SAP Cloud ALM project.
Incorrect e-mail addresses can lead to errors during synchronization. Values that do not contain an "@" sign are treated by the flows as **"no valid e-mail"** and are skipped, which means that no user mapping will be applied for these entries.

5. Final Words and Feedback

Thanks for reading through this documentation and installing the integration content in your Integration Suite. Since this content will evolve in its functionality from time to time we would love to hear from you. Please do not hesitate to give us some feedback whether you like the content or not. In case of missing features we are happy to hear your ideas. We will examine if we can integrate it in the future.

Appendix

Payload Based Mapping (double click):