User Guide | PUBLIC
Document Version: 2204a – 2022-04-19

Actions and Notifications

© 2022 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

Content

1 Actions and Notifications: Overview. 3

1.1 Create an Action. 4

Service Destinations. 11

List Processing. 13

Date and Time Conversion. 16

1.2 Maintain an Action. 17

1.3 Delete an Action. 18

1.4 Search for an Action. 19

1.5 Notification and Email Setup. 19

Set up a Notification Service. 20

1.6 Delete Personal Data. 21

1.7 Action-Related Error Codes. 21

2 Backend Integration Overview. 30

2.1 CPI iFlows and Setup. 31

Pre-delivered CPI iFlows. 31

2.2 Workflow Integration. 34

1 Actions and Notifications: Overview

General information on setting up actions and notifications for rule-based thing monitoring

Introduction

In SAP IoT, actions can be used for the following purposes:

- Action as receiver of an event triggered by a rule: In the definition of a rule, an action can be specified that shall be triggered if the rule condition is fulfilled.
- Action as a decision support option: In the definition of a recommendation to solve a particular business situation, one or more actions can be assigned as possible alternatives for the person in charge to solve the situation.

During rule definition, you take care of different system conditions and situations you want to react to. Rule modeling, thus, is dealing with setting up a formal model that reflects the occurrence of such system events and statuses.

However, a rule alone does not specify what shall happen in response to a rule condition that has been fulfilled. To accomplish this, you set up Actions and associate them with a rule that shall be used as a trigger for these actions.

Likewise, in a decision support scenario where the system cannot automatically determine which action to take, you specify various actions from which a user may choose the one that is likely to solve a given business situation best.

> **i** Note
>
> The maximum frequency for the system to trigger an event for a given thing monitored by a streaming rule is one event per 10 seconds. In other words, no more than six events per minute can be triggered.
>
> However, should your use case require a higher rate of events, we encourage you to get in touch with SAP to see if the threshold can be adjusted to your individual needs.

Related Information

Rules: Overview
Decision Support: Overview

1.1 Create an Action

Process steps for creating an action

Context

An action can be triggered by a rule or an existing action. For example, you might create an action to generate a sales order or a service ticket when certain rule conditions are met. Likewise, an action can be triggered by an existing action. For example, you might get an e-mail notification informing about the newly generated sales order.

Actions can perform a variety of activities, like executing HTTP-based APIs, sending e-mails, or creating notifications that can be displayed in Fiori Launchpads.

> ⚠ Caution
>
> When calling service methods with an action, you should avoid using the unsecure HTTP protocol. Instead, to ensure secure data transmission, SAP highly recommends using the HTTPS protocol for accessing the service methods by an action.

It depends on the selected Action type as to which activity should be performed.

Dynamic Content/Action Vocabulary

Actions support dynamic content. Most of the text input fields in the Action UI support placeholder tokens whose values are being replaced at runtime with actual values.

For example, an e-mail subject may contain the token **${thing.name}**, which at the action processing time, is being replaced with the actual name of the Thing that is in scope of this action.

Supported sources of values are:

- Thing Information: Thing ID, External ID, Name, Description, and so on.
- Thing Basic Data: Properties that appear under the *Basic Data* tab in the Thing Modeler UI.
- Thing Measured Values: Properties that appear under the *Measured Values* tab in the Thing Modeler UI.
- Thresholds: If a thing type contains a property set type with thresholds enabled, the threshold values can also be used as a source of value in an action.
  For more information, see the *Thresholds* section in Additional Settings for Numeric Values.
- Custom Master Data associated to the Thing: Data associated to a thing via Custom Master Data API
  See Custom Master Data for details on how to create and associate Custom Master Data to a thing type and thing.
- HTTPS API responses of other actions: Selected elements of HTTPS API responses from other actions can be made available to the current action.
- Other Information: Action name, Rule ID, current time, and so on.

The availability of tokens in a particular action depends on the definition of the thing type associated with the Action.

A list of available tokens appears by entering **${** followed by any character in a dynamic content-enabled text field. For example, typing **${t** opens a popup with the first 10 tokens containing a **t**.

→ Recommendation

Since there could be more than 10 tokens containing a small number of characters, we recommend typing as many characters as known.

i Note

There are further use cases for dynamic content in action text fields:

- Iteration over long lists of items to be processed by an action. For more information on this topic, see List Processing [page 13].
- Rendering and conversion of date and time values. For more information, see Date and Time Conversion [page 16].

Procedure

1. On the launchpad, select the *Actions* tile.
2. On the *Actions* page, choose *New* to create a new action.
3. Enter the name of the action in the *Name* field.
4. Enter the description of the action in the *Description* field.
5. In the *Triggered By* field, select the element type by which the action shall be triggered.
6. Depending on your decision in the previous step, proceed as follows:

Triggered By	Description/User Action
Event from Action	Select the desired action from the *Action* dropdown list.
Event from Rule	Select the desired rule from the *Rule* dropdown list.
Decision Support	Enables the assignment of the action as a possible action for a decision support recommendation. Select a thing type to which the action shall be related.
Custom Event	Lets the action respond to a customer-specific event that has been created via the event configuration service. Select event type, thing type, and the condition under which the event shall trigger the action.
Geofence Event	Enables the assignment of the action to a Geofence event, such as entering or leaving an area. Select a thing type to which the action shall be related.

Based on the selected *Action* or *Rule*, the system populates the *Thing Type* list.

7. Select the *Action Type*.
   a. *Action Type*: Service Integration

Field	Description
Action Type	To configure the destinations, choose *Service Integration*.

Field	Description
Destination	Choose the destination for the action. **i Note** The *Destination* field shows the *name* of the destinations that have been preconfigured in the Destination Service. For more information, see Service Destinations [page 11].
URL	The URL is displayed based on the destination selected.
Invocation Type	Choose the applicable option: ○ *Manual*: When the action is invoked, it remains in *Pending* status until it's executed manually. ○ *Auto*: When the action is invoked, it's executed immediately.
Method	Select the HTTP method: *GET*, *POST*, or *PUT*.
Request body	Enter the payload required by the service method. Payload templates are available for the SAP CPI iFlows that are available with SAP IoT. **i Note** This field supports dynamic content. You must customize the request body once you've inserted the payload template. The maximum size allowed for the HTTP request body is 5000 characters.
Trigger subsequent event	Enable this checkbox if the current action should generate an event that could trigger other actions. With this option enabled, the system displays the *Response Payload Type* dropdown list.
Response Payload Type	Select the required response payload type: ○ *XML* ○ *JSON* This selection is needed in case the content of the service method response for the current action should be available as dynamic content in other actions.

Field	Description

Event Parameters	Define elements of the service method response as dynamic content that is available in other actions.
	
	Choose *New* to add an event parameter. Enter information as follows:
	○ *Name*: Enter a name to identify the parameter.
	○ *Response Payload Path*: Add the response payload path.
	
	In the following example (valid for the *XML* payload type), the event parameter with the response payload path `/RETURN/item/MESSAGE_V1` and name PREQ_NUMBER would make the number of the purchase requisition (0010000338) available as input parameter with the name PREQ_NUMBER of a subsequent action.
	
	`<RETURN>`
	`  <item>`
	`  <TYPE>I</TYPE>`
	`  <CODE>06402</CODE>`
	`  <MESSAGE>Purchase requisition number 0010000338 created</MESSAGE>`
	`  <LOG_NO/>`
	`  <LOG_MSG_NO>000000</LOG_MSG_NO>`
	`  <MESSAGE_V1>0010000338</MESSAGE_V1>`
	`  <MESSAGE_V2/>`
	`  <MESSAGE_V3/>`
	`  <MESSAGE_V4/>`
	`  </item>`
	`</RETURN>`
	
	Choose *Save*.

b. *Action Type*: Select *Email Notification*.

Field	Description
Action Type	Select *Email Notification* to create an action for which you can configure an e-mail.
Recipients	Enter the e-mail address of the recipient. You can enter multiple addresses, separated by semicolons. **i Note** This field supports dynamic content.
Subject	Enter the subject of the mail. **i Note** This field supports dynamic content.

Field	Description
Text	Enter the text of the mail. **i Note** This field supports dynamic content. **i Note** The maximum size allowed for the mail body is 5000 characters.

c. *Action Type*: Select *In-App Notification*.

Field	Description
Action Type	Select *In-App Notification* to create an action for which you can configure a notification.
Recipients	Enter the SAP IoT logon name of the recipient. You can enter multiple recipients, separated by semicolons. **i Note** This field supports dynamic content.
Text	Enter the text of the notification message. **i Note** This field supports dynamic content. **i Note** The maximum size allowed for the In-App body is 5000 characters.

In-App Notifications can be configured to be actionable. When you click the In-App Notifications in the Fiori Launchpad, the browser navigates to the target object and target action and passes on the target parameters defined in the Action Modeler UI.

Field	Description
Target Object	The navigation target object (for example, *Purchase Order* application)
Target Action	The action of the navigation target object (example, View)
Target Parameters	Define parameters that are passed along to the navigation target object and action (for example, the ID of the purchase order that should be viewed)

d. *Action Type*: Decision Support

Field	Description
Action Type	To configure the destinations, choose *Decision Support*.
Alias	To link the decision support information to the action, enter a name in the *Alias* field. **i Note** Make sure that the name in the *Alias* field matches with the name in the *Action* field.
Payload	Enter the payload required by the service method. Payload templates are available for the SAP CPI iFlows that are available with SAP IoT. **i Note** This field supports dynamic content. You must customize the request body once you've inserted the payload template. The maximum size allowed for the HTTP request body is 5000 characters.

e. *Action Type*: Event Mesh

The SAP Event Mesh service (formerly known as Enterprise Messaging System) (EMS) allows you to specify the message payload to be created and also lets you choose the message channel, that is, Queues or Topics. For more information on SAP Event Mesh, see What is SAP Event Mesh and Initial Setup respectively.

For more information on Messaging Channels, see Messaging Concepts.

Field	Description
Action Type	To configure the destinations, choose *Event Mesh*.
Destination	Choose the destination for the action. **i Note** The *Destination* field shows the *name* of the destinations that have been preconfigured in the Destination Service. For more information on the required settings for the assigned destination, see section *Defining a Destination for an Action of Type Event Mesh* in Service Destinations [page 11].

Field	Description
Message Channel	Choose the Message Channel option: ○ *Topics*: When the action is invoked, the service enables a sending application to publish the message to a topic. ○ *Queues*: When the action is invoked, the service enables a sending application to publish the message to a specific queue.
Topic Name or *Queue Name*	Depending on your choice in the previous step, proceed as follows: **Triggered By** / **Action** *Topic Name* / Enter a name for the topic. *Queue Name* / Enter a name for the queue.
Message	Enter the message for the action. **i Note** This field supports dynamic content. **i Note** The maximum size allowed for the In-App body is 5000 characters.
Additional Properties	Enter the following property name and value: **Name** / **Value** `sap.enterprise_messaging` / `true` **i Note** The above property first needs to be configured for the respective destination (for example, via SAP BTP Cockpit), that is, **not** within SAP IoT. Also, activate *Use Default JDK Truststore*.

8. Choose *Save*.

Results

The new action is created.

Related Information

Service Destinations [page 11]
List Processing [page 13]
Date and Time Conversion [page 16]

1.1.1 Service Destinations

Actions of type Service Integration use the destinations that are defined in the SAP BTP Destination Service.

Those destinations define how to access the endpoints (for example, SAP systems) which are the targets of those actions. Hence, actions of type Service Integration must have one of these destinations assigned.

Creating Destinations

Destinations can be created in the following ways:

- Via the Destination Service UI of SAP BTP Cockpit: In the cloud platform cockpit, within the menu on the left, *Connectivity* → *Destinations* → *New Destination* (certain roles and permissions are needed to create a destination from the UI).
- Via the Destination Service API provided by SAP BTP.

Supported Destination Type and Authentication Mechanisms

Destinations have to be of type HTTP and support multiple, but not all, authentication types. Supported authentication types are:

- NoAuthentication
- BasicAuthentication
- OAuth2ClientCredentials (Client Credentials and Password grant types are supported.)

Supported Destination Proxy Types

Destinations can describe Cloud systems (reached via public Internet) or On-Premise systems (reached via SAP Cloud Connector). Depending on the location of the destination, the proxy type has to be defined accordingly:

- Internet
- On-premise

Supported Destination URLs

Destinations of type HTTP need to have a URL. The URL can have tokens, which are replaced with actual values during the execution of an action. These tokens are optional.

Since destinations are defined outside of the Action UI, there's no value help available for those tokens. The token format is still the same though: **${token.name}**.

An example with the thing ID, name, and type tokens in a URL looks like this: `https://www.sap.com/do/${thing.type}/something?thingid=${thing.id}&thingname=${thing.name}`

Destinations with HTTP Headers

HTTP Headers can be defined using (additional) destination properties. In the destination service UI, a property can be added by clicking the *New Property* button; the left field being the property name, and the right being the property value. Properties can also be defined via the destination service API.

To be used as an HTTP Header, the property must have the prefix **sap.iot.header.** (including the last dot) followed by the name of the actual header variable.

Example: sap.iot.header.**X-Client-Version** or sap.iot.header.**X-Client-ID**

Destinations for Endpoints Which Require a CSRF Token

SAP systems often require a CSRF (cross-site request forgery) token.

This protection can often be disabled, but in case this is not appropriate or possible, the destination definition can be enhanced to indicate that a CSRF token needs to be retrieved and added to the request.

The two properties needed for that are:

- **sap.iot.fetchXcsrf** with the value **true**
- **sap.iot.XcsrfURL** with the value of the URL, which returns the CSRF token. This is often the same URL as the actual URL of the endpoint.

Destinations for Endpoints with a Self-signed Certificate

Although not considered secure, there might be a need to use the action framework to connect to systems that are using self-signed certificates (for example, demo or development systems).

The property **sap.iot.certificationEnabled** with the value **false** can be used in a destination to indicate that the certificate check does **not** apply for this destination.

Defining a Destination for an Action of Type *Event Mesh*

For actions of type *Event Mesh*, the assigned destination must have been set up as follows:

Name	Provide the destination name of the *Event Mesh* service instance.
Client ID, *Client Secret*, *URL*, *Token Service URL*	Use the values of the service keys defined in the previous setup.
Additional Properties	Create a name/value pair as follows: `sap.enterprise_messaging: true`

Finally, make sure *Use default JDK truststore* is checked.

1.1.2 List Processing

Including long lists of items in action text fields.

Introduction

When defining actions, you may find yourself in situations where you need to process or inform about (or both) comprehensive lists of items. Here are some examples of such lists:

- Bill of material
- Sales order
- List of ingredients
- Warehouse inventory

In principle , it would be possible to enumerate all the items of such a list in the respective part of an action (request payload, body of an email or in-app alert). However, doing so could lead to different kinds of problems:

- Wrong entries as a typical side effect of any tedious and boring task.
- In case of very large lists, the size limits of the text fields used for action configuration might be exceeded.

To cope with these challenges, the *Action Modeler* of SAP IoT provides a feature for addressing all the items in a given list with minimum effort and space needed.

Details

To make addressing large amounts of items easier at design time, *Action Modeler* offers a special syntax to be used in certain text fields. The basic approach is to include a reference to a particular container-like entity of which you know it represents a list of equally structured items. The syntactical tags used for the container entity are then indicating that the system shouldn't deal with the container entity, but with the items it contains. Actually, the system performs a loop over all the items found in the container and processes them as desired.

> **Example**
>
> You want to define an action where a number of processing steps shall be executed for each month of the year. You've defined an array variable named `month`, which contains a list of all the months of the year, that is `[January,February,March,…]`. With the list processing syntax, all you have to do is to include a reference to the `month` array to instruct the system to expand that variable to the individual values it contains.

Syntax

The following syntax elements are available for list processing:

List Processing Syntax Elements

Annotation	Explanation
`<#list>...</#list>`	The `<#list>` tag is used for listing a sequence or collection of items. It simply outputs all the items of the content enclosed by the tags (for example, a collection of event data, an array of Strings like [January, February, March], etc.). **Example** `<#list> "month": ${month} </#list>`

`<#sep>...</#sep>`	(Optional) Used in conjunction with the <#list> tag. Indicates that the text after the tag shall be used as a separator between each list item, except for the last item.
	
	When used immediately before the closing </#list> tag, there's no need to enter the closing </#sep> tag. Otherwise, the <#sep> tag requires a closing tag.
	
	Example
	`${month} = [January, February, March]`
	• Usage without closing </#sep> tag:
	`<#list> ${month}<#sep>, </#list>`
	In this example, a comma followed by a blank is defined as a separator for each month found in the `${month}` variable.
	Result: January, February, March
	Note that the last element "March" is **not** followed by the specified separator.
	• Usage with closing </#sep> tag required:
	`<#list> ${month} ${day} <#sep>,</#sep> 2021 </#list>`
	Result: January 1, 2021 January 2, 2021 January 3 2021
	○ In this example, only the comma is used as a separator but not the trailing year "2021". Including "2021" in the separator tag would lead to "2021" missing for the last entry.
	○ Thus, "3" is **not** followed by a comma, because it's part of the last item in the collection. However, it does still contain "2021".

Note

The two given annotations for list processing are the only allowed tags starting with "<#" within the definition of an action. Any attempt to enter something different from either "list" or "sep" results in an error.

Output Rendering

When using list processing annotations in the payload of an action, the positioning and arrangement of these elements has a direct impact of the output rendering. To illustrate that, compare the following examples:

Example

- `<#list> ${month}<#sep>, </#list>`
  Result: January, February, March
- ```
  <#list> ${month}<#sep>,
  </#list>
  ```

  Result:
  January,
  February,
  March
  Note that in this second example, the separator in the payload is followed by a line feed, which is perfectly reflected in the generated output with each list item rendered in a new line.

Payload Example

The following example illustrates an action payload used to iterate through the items of a purchase requisition:

Source Code

```
{
    "PurchaseRequisition": {
        "eventId": "${event.id}",
        "RequisitionItems": [
            <#list>
            {
                "productId": "${Product.ProductId}",
                "purchaseOrderItems": "${Product.ProductName}",
                "qualityCheckMaterial": "${PurchaseOrderItem.Material}",
                "purchaseOrderQuant": "${ScanEvent.scannedQuantity}"
            }<#sep>,
            </#list>
        ]
    }
}
```

At runtime, this payload is resolved to the following result (JSON):

Output Code

```
{
    "PurchaseRequisition": {
        "eventId": "EAF69F6DFF6E46EAB5467F2BB281294C",
        "RequisitionItems": [
            {
                "productId": "WP8182528",
                "purchaseOrderItems": "Dryer Heating Element",
                "qualityCheckMaterial": "WP8182528",
                "purchaseOrderQuant": "3"
            },
            {
                "productId": "WP1084394",
                "purchaseOrderItems": "Dryer Thermostat",
                "qualityCheckMaterial": "WP1084394",
                "purchaseOrderQuant": "2"
```
```

```
                }
            ]
        }
}
```

1.1.3 Date and Time Conversion

How to convert and render date and time values

Introduction

There are many situations in daily business where date and time values are important. Here's a small list of examples:

- Start and end times of all kinds of processes
- Processing duration
- Order and delivery dates
- Intervals of recurring events
- Time window for sales promotion activities
- Reception time of an alarm notification

Any of these business events may be related to an action in SAP IoT, be it as a timestamp of the action itself, or as a date and time information, which is reported by the action. However, reporting about date and time values is subject to localization and therefore requires adaptation of the raw values to the expectations of the target audience. Otherwise, all kinds of misconceptions can arise with potentially dramatic consequences.

To fulfill the audience expectations with regards to the presentation of date and time values, the action component of SAP IoT offers a powerful function for converting and rendering date and time values in many different ways. You can use this feature in the various text fields used in actions (for example, e-mail subject or body, or the respective fields used for in-app alerts).

Details

To use the date and time conversion feature, insert the following element in the text field of an action:

```
^{dateConverter(${<TimeToken>},"<DateTimePattern>","<TimeZoneID>")}
```

For the parameters, see the following table:

Parameter	Description
`TimeToken`	Mandatory. Any field or variable representing a date and time value, and that the action can access in a given business scenario (for example, `event.time`, `BusinessTime`)

Parameter	Description
`DateTimePattern`	Mandatory. Formal representation of the different parts of which a date and time value consists. For an overview and examples, see the following page on GitHub: DateTime format patterns .
`TimeZoneID`	Optional. ISO abbreviation of the time zone (for example, CET, PST, EST, IST, CST, AST). If this parameter is omitted, the time zone defaults to UTC.

> **i Note**
>
> Make sure that the parameters passed to the `dateConverter` function are separated by commas only. That is, there are no spaces allowed inside of the parentheses.

Examples

Example 1

```
^{dateConverter(${event.time},"yy-mm-dd HH:mm:ss.SSS")}
```

In this example, the time when an event occurred is given with a precision of 1 millisecond. As the timezone parameter has been omitted, the time is given as UTC.

Example 2

```
^{dateConverter(${order.date},"dd-mm-yy HH:mm:ss","CET")}
```

In this example, the date and time when a purchase order has been received by a supplier is returned using double-digit date format (for example, "02. 04. 2020"). The values refer to Central European Time (CET).

Example 3

```
s = "The order placed on ",^{dateConverter(${order.date},"yy-mm-dd
HH:mm:ss","CET")}, " will be fulfilled by delivery on ", ^{dateConverter($
{shipment.date},"yy-mm-dd HH:mm:ss","CET")}, "."
```

In this example, a string variable is defined using two different calls to the `dateConverter` function (one for an order, the other for a shipment).

1.2 Maintain an Action

Process steps for modifying an action

Context

This feature allows you to update or modify an existing action.

> **Note**
>
> For more detailed information on the various settings of an action, see Create an Action [page 4].

Procedure

1. On the launchpad, select the *Actions* tile.
2. On the *Actions* page, click the link of the action to be modified, to navigate to the Action detail page.
3. Change the content of the field on the selected action page.
4. Select the required options from the *Trigger by* and *Action Type* fields.
5. Enter the updated details in the HTTP, Email, In-App Information, Decision Support Information, or Enterprise Messaging System Information on the selected *Action Type*.
6. Choose *Save*.

Results

The changes are applied to the selected action.

1.3 Delete an Action

Process steps for deleting an action

Context

This feature allows you to delete an existing action.

> **Note**
>
> One or more actions can be selected and deleted.

Procedure

1. On the launchpad, select the *Actions* tile.
2. On the *Actions* page, enable the check box against the action to be deleted.

3. Choose *Delete*.

Results

The selected action is deleted.

1.4 Search for an Action

Process steps for searching an action

Context

This feature allows you to search for a specific action. In addition to obvious search criteria such as *Name* or *Type* of an action, you can also filter the list for actions that are associated with a particular *Thing Type* or that are *Triggered By* a particular action or rule.

Procedure

1. On the launchpad, select the *Actions* tile.
2. On the *Actions* page, enter the name of the Action you want to search for, or enter the applicable search string.
3. Select *Service Integration*, *Email Notification* or *In-App Notification* from the *Action Type* dropdown menu.
4. Select the applicable rule or action from the *Triggered By* dropdown menu.
5. Choose *Go*.

   The page displays all the actions that match the search criteria.
6. Choose *Clear* to clear all the search strings and search results.
   The system then repopulates the list of actions with all available actions, and you may start a new search.

1.5 Notification and Email Setup

This feature allows you to configure the mail server and to set up notifications and email alerts.

Before you can start sending email alerts, you first have to provide the necessary information about your email infrastructure (such as server names and ports to be used). After that, you can define actions of type *Email Notification*.

Once you are done with the setup, the system will send email alerts as well as in-app notifications every time a rule condition is fulfilled and that rule is associated with an action that sends emails or in-app notifications. Emails as well as in-app notifications are kept in the system for 30 days after the triggering event for reference purposes. After that period, emails and notifications are automatically deleted by an internal cleanup job.

> **i** Note
>
> After having sent an email, the system waits for a response from the receiving mail server. If the receiving mail server does **not** send a response code indicating successful delivery, the notification service resends the mail. However, attempts to resend the mail are made only within the first three minutes after the first sending attempt. After that, the mail is considered as undeliverable.

1.5.1 Set up a Notification Service

Process steps for configuring a new mail server

Context

This feature allows you to set up a new mail server configuration.

> **i** Note
>
> The mail server has to be configured first, before an email can be sent.

Procedure

1. To set up a new mail server configuration, choose *Mail Server Configuration*.
2. Enter the *SMTP Server Host* address.
3. Enter the *SMTP Server Port* detail.
4. Enter the *SMTP Server Authentication User* name.
5. Enter the *SMTP Server Authentication Password*.
6. Enter the email address of the notification sender in the *Notification Sender Email Address* field.
7. Enter the default subject of the notification in the *Default Notification Address* field.
8. Choose *Save*.

1.6 Delete Personal Data

In order to delete personal data, you have to use the list page to locate the actions with type EMAIL or INAPP. Open those actions and remove the e-mail address (EMAIL Action) or the Cloud Foundry login name (INAPP) of the person whose data needs to be deleted from the list of recipients.

> **i** Note
>
> Since the input fields for both e-mail address and login name are validated by the system, you cannot simply delete existing personal data from these fields because the system wouldn't allow saving an action with invalid or missing data in a mandatory field. Instead, enter some anonymized data that matches the formal requirements for the field in question (e.g., you might enter `some.person@company.com` as a dummy e-mail address and then save the action).

1.7 Action-Related Error Codes

Overview of all types of errors that can occur with action processing.

In this section, you can inform yourself about the different types of errors that can occur when the system processes an action. Note that many of these error types can only occur when you're programming against the respective API services. As opposed to that, when you define actions via the Action Modeler apps delivered by SAP, these apps take care of the necessary proper format settings or the allowed values for certain properties of an action.

> **i** Note
>
> When looking at the list of error codes, don't get confused about any of the following observations:
>
> - The list of error codes is **not** a gapless sequence of numbers. This is with intent and **not** a mistake.
> - Several errors are very similar or even identical. However, the unique error code of each error helps our support experts to determine the source code location where the error occurred.
> - For a number of error situations, there is no way for customers to correct the situation. For these cases, the suggested solution is "Create a support ticket." The proper component for all errors listed here is `IOT-BSV-ACT`.

Action-Related Error Codes

Error Code	Category	Root Cause	Solution
`iot.actions.error.1`	Invocation Type Missing	An action requires an invocation type, which can either be “AUTO” or “MANUAL”. An empty type is not allowed.	Specify the invocation type of the actions.
`iot.actions.error.2`	Wrong Invocation Type	An invocation type other than "AUTO" or "MANUAL" has been specified for an action .	Make sure that the invocation type is correctly specified.

Error Code	Category	Root Cause	Solution
`iot.actions.error.3`	Action Processing Failed	n/a	Create a support ticket.
`iot.actions.error.4`	Action Processing Failed	Trigger event for the action could not be processed.	Create a support ticket.
`iot.actions.error.6`	Action Type Wrong or Missing	An unsupported action type has been specified, or no action type at all. This is not allowed.	Make sure that the action type is correctly specified as one of the following: **`INAPP`**, **`EMAIL`**, **`HTTP`**.
`iot.actions.error.7`	Runtime Exception	Error occurred while processing an action.	Create a support ticket.
`iot.actions.error.8`	Runtime Exception	Action or event has been deleted while the action was executed.	Create a support ticket.
`iot.actions.error.16`	User Error	Recipient missing for e-mail action	Provide a valid e-mail address for the mail recipient.
`iot.actions.error.17`	User Error	Subject missing for e-mail action	Provide a subject for the e-mail.
`iot.actions.error.18`	User Error	Content missing for *Body* field of e-mail or in-app actions.	Provide some content to help end users solve the situation.
`iot.actions.error.19`	Runtime Exception/Notification Error	Exception occurred while sending a notification.	Create a support ticket.
`iot.actions.error.20`	User Error	Request body of HTTP action is empty.	Provide a valid request body.
`iot.actions.error.21`	Runtime Exception/Run ID Generation	Proper run ID could not be generated.	Create a support ticket.
`iot.actions.error.22`	Runtime Exception/Run ID Generation	Error occurred while processing chained actions.	Create a support ticket.
`iot.actions.error.23`	Runtime Exception/ Parameter Processing	Unable to find proper parameter sources for action processing.	Create a support ticket.
`iot.actions.error.24`	Runtime Exception/ Parameter Processing	Unable to find proper parameter paths for action processing.	Create a support ticket.
`iot.actions.error.25`	Runtime Exception/ Parameter Processing	Invalid combination of parameter path and source.	Create a support ticket.
`iot.actions.error.26`	Runtime Exception/Notification Error	Sending of in-app notification failed.	Create a support ticket.
`iot.actions.error.32`	Runtime Exception/PST Actions	Unable to find named property set for actions based on property set types (PST).	Create a support ticket.
`iot.actions.error.33`	Runtime Exception/PST Actions	No thing ID found while processing the actions.	Create a support ticket.

Error Code	Category	Root Cause	Solution
`iot.actions.error.34`	Runtime Exception/PST Actions	Processing failed: Neither thing type nor property set type provided.	Create a support ticket.
`iot.actions.error.35`	Runtime Exception/PST Actions	No proper parameters found for action based on property set type.	Create a support ticket.
`iot.actions.error.36`	Runtime Exception/PST Actions	Action references both thing type and property set type. This is not allowed.	Create a support ticket.
`iot.actions.error.48`	User Error/E-Mail Action	E-mail address missing.	Provide a valid e-mail address.
`iot.actions.error.49`	User Error/E-Mail Action	Invalid e-mail address.	Provide a valid and well-formed e-mail address (for example, **`name@company.com`**).
`iot.actions.error.50`	Runtime Exception	Illegal access exception occurred while extracting token.	Create a support ticket.
`iot.actions.error.53`	Runtime Exception/ Metadata	Retrieving metadata value for a thing failed.	Create a support ticket.
`iot.actions.error.54`	Runtime Exception/Metadata	Metadata value returned for a thing was null.	Create a support ticket.
`iot.actions.error.55`	Runtime Exception/ Parameter Processing	No parameter found for the given source to process the actions. Note: Parameters are the tokens that you type while creating an action while source refers to the parameter type (for example, `TS` for time series).	Create a support ticket.
`iot.actions.error.56`	User Error/Master Data	Error occurred while processing business partner data.	Provide proper business partner values. For details, see Modeling Custom Master Data Overview.
`iot.actions.error.57`	User Error/Master Data	Unknown parameters for business partner.	Provide proper business partner values and parameters. For details, see Modeling Custom Master Data Overview.
`iot.actions.error.59`	User Error/Master Data	The provided subtype is not supported for given master data type.	Provide proper subtype for given master data type. For details, see Modeling Custom Master Data Overview.
`iot.actions.error.60`	User Error/Master Data	Custom master data property `Name` not found.	Check master data for correctness. If you can't find any error, create a support ticket.

Error Code	Category	Root Cause	Solution
`iot.actions.error.61`	Runtime Exception/ Master Data	Runtime exception occurred while processing master data properties.	Create a support ticket.
`iot.actions.error.62`	Runtime Exception/ Master Data	Invalid values returned while retrieving master data associations for a thing.	Create a support ticket.
`iot.actions.error.63`	Runtime Exception/ Master Data	No value returned while retrieving master data associations for a thing.	Create a support ticket.
`iot.actions.error.64`	User Error/Master Data	Business partner instance found without ID.	Provide proper master data values. For details, see Modeling Custom Master Data Overview.
`iot.actions.error.65`	User Error/Master Data	No business partner instance found from master data associations for business partner.	Provide proper master data values. For details, see Modeling Custom Master Data Overview.
`iot.actions.error.67`	Runtime Exception/ Master Data	Error parsing master data association values.	Create a support ticket.
`iot.actions.error.68`	Runtime Exception/ Master Data	Problem detected in validating associations for master data.	Create a support ticket.
`iot.actions.error.69`	Runtime Exception/ Master Data	Error retrieving the detailed master data values.	Check master data associations for correctness. If you can't find any error, create a support ticket.
`iot.actions.error.70`	User Error/Master Data	Parameters not valid for master data source type.	Check that master data is properly defined.
`iot.actions.error.71`	User Error/Master Data	Retrieved empty object instance association for master data.	Check that object type for master data is properly defined.
`iot.actions.error.72`	Runtime Exception/ Master Data	No values returned by master data APIs	Create a support ticket.
`iot.actions.error.73`	Runtime Exception/ Master Data	Empty object returned by master data APIs.	Create a support ticket.
`iot.actions.error.74`	Runtime Exception/ Master Data	The metadata values for master data properties are empty.	Create a support ticket.
`iot.actions.error.75`	Runtime Exception/ Master Data	Exception occurred while parsing the JSON response for master data properties.	Create a support ticket.
`iot.actions.error.76`	User Error/Master Data Notification	No non-token recipient and no instance object found for master data.	Check that the master data instance objects are defined properly.
`iot.actions.error.77`	User Error/Master Data Notification	No non-token recipient and master data recipients found.	Check that the master data recipients are defined properly.

Error Code	Category	Root Cause	Solution
iot.actions.error.78	Runtime Exception/Database	Required fields to save action result not found.	Create a support ticket.
iot.actions.error.79	Runtime Exception/Database	Action results not saved due to database error.	Create a support ticket.
iot.actions.error.80	Runtime Exception/Action Processing	Message broker fails to send events for the actions.	Create a support ticket.
iot.actions.error.83	Runtime Exception/ Master Data	Error retrieving values from business partner URL.	Create a support ticket.
iot.actions.error.84	User Error/Master Data	Business partner not defined.	Check that master data is properly defined.
iot.actions.error.85	Runtime Exception/Action Processing	Error retrieving value from snapshot API.	Wait for retry. If the action still fails, create a support ticket.
iot.actions.error.86	Runtime Exception/ Master Data	Error parsing JSON data for association.	Create a support ticket.
iot.actions.error.87	Runtime Exception/ Master Data	Data association URL is empty.	Create a support ticket.
iot.actions.error.100	User Error/Destination	Destination details missing or invalid.	Ask your system administrator.
iot.actions.error.101	User Error/Destination	Destination URL missing.	Ask your system administrator.
iot.actions.error.102	User Error/Destination	Destination authentication missing.	Ask your system administrator.
iot.actions.error.103	User Error/Destination	Destination user missing.	Ask your system administrator.
iot.actions.error.104	User Error/Destination	Destination password missing.	Ask your system administrator.
iot.actions.error.105	User Error/Destination	Destination client ID missing.	Ask your system administrator.
iot.actions.error.106	User Error/Destination	Destination client secret missing.	Ask your system administrator.
iot.actions.error.107	User Error/Destination	Token service URL missing for destination.	Ask your system administrator.
iot.actions.error.110	User Error/Destination	Token service user missing for destination.	Ask your system administrator.
iot.actions.error.111	User Error/Destination	Token service password missing for destination.	Ask your system administrator.
iot.actions.error.112	User Error/Destination	Unsupported authentication type for destination.	Ask your system administrator.

Error Code	Category	Root Cause	Solution
`iot.actions.error.113`	User Error/Master Data	Retrieving master data association failed: Thing ID missing.	Provide the thing ID.
`iot.actions.error.114`	User Error/Master Data	No proper composite master data URL found.	Provide the correct thing ID.
`iot.actions.error.115`	User Error/Master Data	Business partner instance found without business partner ID.	Provide a correct business partner ID.
`iot.actions.error.116`	User Error/Master Data	No basic data found for business partner.	Provide a correct business partner ID.
`iot.actions.error.117`	User Error/Master Data	Business partner application name is null.	Provide a correct business partner ID.
`iot.actions.error.118`	User Error/Master Data	Business partner ID is null or empty.	Provide a correct business partner ID.
`iot.actions.error.120`	Runtime Exception/Acion Processing	Error retrieving value for time series parameters; service may be down.	Wait for retry. If the action still fails, create a support ticket.
`iot.actions.error.121`	Runtime Exception/Acion Processing	Call to external endpoint failed.	Wait for retry. If the action still fails, create a support ticket.
`iot.actions.error.122`	User Error/Action Processing	HTTP method is null.	Provide a proper HTTP method (for example, `GET` or `POST`).
`iot.actions.error.123`	User Error/Action Processing	Body of HTTP GET request contains data.	Check the request (payloads for GET requests cannot be processed).
`iot.actions.error.125`	User Error/Action Processing	No response parameter found for HTTP call.	Define proper response parameter for chained actions. See Create an Action [page 4].
`iot.actions.error.126`	User Error/Action Processing	Response document either missing or of wrong type.	Make sure that the response type is either JSON or XML.
`iot.actions.error.127`	Internal Error/Action Processing	Parameters not found for processing actions for a particular source.	Create a support ticket.
`iot.actions.error.128`	User Error/Chained Action Processing	No saved result found for last action run.	Check the definition of the chained actions.
`iot.actions.error.129`	User Error/Chained Action Processing	Saved result for last action run is empty or null.	Check the definition of the chained actions.
`iot.actions.error.146`	Internal Error/Event Processing	Action-based event without action reference.	Create a support ticket.
`iot.actions.error.147`	Internal Error/Event Processing	Unexpected event type.	Create a support ticket.
`iot.actions.error.148`	Internal Error/Event Processing	Event element "data → thing ID" missing or invalid.	Create a support ticket.

Error Code	Category	Root Cause	Solution
iot.actions.error.149	Internal Error/Event Processing	Property set value missing for action based on property set type.	Create a support ticket.
iot.actions.error.150	Internal Error/Action Processing	Invalid action type.	Create a support ticket.
iot.actions.error.151	User Error/Action Processing	Invalid X-CSRF or ETag.	Provide a proper URL for X-CSRF or ETag.
iot.actions.error.152	User Error/Action Processing	Failed to retrieve X-CSRF token.	Provide a proper URL for X-CSRF,
iot.actions.error.153	User Error/Action Processing	Failed to retrieve ETag token.	Provide a proper URL for ETag.
iot.actions.error.154	User Error/Action Processing	Failed to perform HTTP request with URL syntax.	Provide a proper URL in actions.
iot.actions.error.155	User Error/Action Processing	HTTP call failed.	Create a support ticket.
iot.actions.error.156	Internal Error/Action Processing	Call to custom master data API failed.	
iot.actions.error.163	Runtime Exception/ Direct-call Action	Direct call event type without target action reference.	Create a support ticket.
iot.actions.error.164	Runtime Exception/ Direct-call Action	Direct call event type without correlation id reference.	Create a support ticket.
iot.actions.error.165	User Error	JSON result parsing error for HTTP response path.	The response payload path of an event parameter in the action is incorrect. Please correct it.
iot.actions.error.166	User Error	XML result parsing error for HTTP response path.	The response payload path of an event parameter in the action is incorrect. Please correct it.
iot.actions.error.170	External Endpoint Error	Call to external endpoint failed.	Check that the external endpoint is reachable.
iot.actions.error.171	External Endpoint Error	Call to external endpoint failed due to timeout.	Check that the external endpoint is reachable.
iot.actions.error.172	External Endpoint Error	Call to external endpoint failed: OAuth token could not be obtained.	Check the OAuth setting of the external endpoint.
iot.actions.error.173	Internal Error/Action Processing	Error retrieving value from EventPropertySetTypes API.	Wait for retry. If the action still fails, create a support ticket.
iot.actions.error.175	Internal Error/Action Processing	Error processing an event.	Create a support ticket.
iot.actions.error.200	Runtime Exception/Action Processing	Exception during API call	Create a support ticket.

Error Code	Category	Root Cause	Solution
`iot.actions.error.201`	Runtime Exception/Date Processing	Error parsing expression: Invalid date pattern.	Create a support ticket.
`iot.actions.error.202`	Runtime Exception/Date Processing	Invalid date token.	Create a support ticket.
`iot.actions.error.203`	Runtime Exception/Date Processing	Error parsing expression: Invalid timestamp.	Create a support ticket.
`iot.actions.error.204`	Runtime Exception/Date Processing	Invalid time zone.	Create a support ticket.
`iot.actions.error.205`	User Error	Invalid thing ID provided by user.	Check and correct the thing Id. Then retry.
`iot.actions.error.206`	Runtime Exception/Processing Usage	Error obtaining actual processing usage of actions.	Create a support ticket.
`iot.actions.error.207`	Runtime Exception/Action Processing	Error obtaining property set type/ property set.	Create a support ticket.
`iot.actions.error.208`	Runtime Exception/Action Processing	`<EventPropertySetType>` missing for action.	Create a support ticket.
`iot.actions.error.209`	Runtime Exception/Action Processing	Error building outgoing content for incoming batch cloud event.	Create a support ticket.
`iot.actions.error.210`	Runtime Exception/Action Processing	Error during posting event.	Create a support ticket.
`iot.actions.error.211`	Runtime Exception/Action Processing	Event property set missing in an action-fired event.	Create a support ticket.
`iot.actions.error.212`	Runtime Exception/Action Processing	Event property sets expected in an event, but missing.	Create a support ticket.
`iot.actions.error.214`	User Error/List Processing	List annotation is not supported in header or body content for action types *In-App* or *Email Notification* for an event with array of events	Delete list annotation from action content.
`iot.actions.error.215`	User Error/List Processing	Event source is missing or the source type is not supported.	Correct the source in the event triggering the action, if possible. Otherwise, create a support ticket.
`iot.actions.error.216`	User Error/List Processing	List annotation is not supported in action for event type with attribute `<isBatch = false>`.	Delete list annotation from action content.
`iot.actions.error.217`	Runtime Exception/ Template Engine	Empty content returned from template engine for action.	Create a support ticket.
`iot.actions.error.218`	Runtime Exception/Validation	Validation of a cloud event failed.	Create a support ticket.

Error Code	Category	Root Cause	Solution
`iot.actions.error.219`	Runtime Exception/Validation	Empty data array in incoming event.	Correct the incoming event triggering the action, if possible. Otherwise, create a support ticket.
`iot.actions.error.220`	Runtime Exception/Validation	Header or body content for action of type *In-App* or *Email Notification* is invalid.	Correct the action content.
`iot.actions.error.221`	Runtime Exception/Action Processing	Unable to find base event type due to missing action guide.	Create a support ticket
`iot.actions.error.222`	Runtime Exception/Action Processing	Unable to find base event type: Action must have exactly one action event in the database.	Create a support ticket
`iot.actions.error.223`	Runtime Exception/Action Processing	Unable to obtain value for `<isBatch>` from `Events` API.	Create a support ticket
`iot.actions.error.224`	User Error	Unable to test action of custom event type with an array of events.	Test function does not support actions triggered by a custom event type having an array of events.
`sap.iot.notifications.01`	Notification Server	Problem occurred with SMTP configuration.	Check the SMTP parameters.
`sap.iot.notifications.03`	Notification Server	Problem sending out a notification.	Check the logs for more details.
`sap.iot.notifications.05`	Notification Server	Invalid notification configuration.	Check the logs for more details.
`sap.iot.notifications.06`	Notification Server	Email expired after the specified time threshold.	Try extending the time threshold for repeated sending attempts (default 3 min).
`sap.iot.notifications.07`	Notification Server	Notification was successfully dispatched.	n/a

2 Backend Integration Overview

Backend integration helps in creating and updating business objects in the SAP backend system so as to trigger it from SAP IoT.

As a key user, you should be able to create a rule with an action that automatically triggers the creation or update of a business object in the SAP business system. The rule action should pass all the relevant device and master data parameters that are required to create or update the remote business object. This enables you to react on the alert notification which enables the "Insight to Action" scenario.

> **Example**
>
> The temperature is >18 and the cooler is powered on for more than 12 hours --> Create a service request once in the CRM or Cloud for Service.

Predefined and configurable iFlows are made available in this release, for the following business objects:

- Creation of Service Tickets (SAP C4C, SAP CRM)
- Creation of Sales Order
- Creation of Purchase Requisition

> **Note**
>
> In addition to the above list, the execution of HTTP-based APIs, for example, SAP CPI iFlow endpoints or customer backend systems are allowed.

The link between the SAP IoT and the backend systems for these scenarios can be established through CPI I-Flows (or other such mechanisms). You should be able to use this mechanism/framework to enable the creation of business objects (irrespective of customizations).

Here, the creation of these business objects can be achieved through an action framework. You should have an API/UI component to define the business object (CPI I-Flow URL, authentication, payload, and so on).

The following parameters (not limited to) need to be configurable:

- The SAP IoT tenant through which you want to perform the integration
- The backend system where the business object shall be created
- Payload and other parameters to be passed for the business object creation (depending on the customization at different customers, and so on)
- Different types of data that can get used (master data, time series data, thing metadata, and so on)
- Protocol followed (HTTP, and so on)
- Type of Call (GET, POST, PUT)
- Authentication
- Notification template

The data about the business object should be persisted and the creation should be logged (business object type, timestamp, associated device, rule associated with creation, created by). The persisted data should be available for consumption in the other parts of the application (KPIs, visualization, reports, and so on). For example, the number of service requests on device X

Auto and manual mode of action (business object creation) execution: Auto mode is to send the action right away when the rule condition is met, manual mode is to give the user the option to reconsider/adjust the action prior to sending it out.

A notification is triggered every time a business object is triggered. The notification should have the details about the business object that was created (business object number, and so on).

2.1 CPI iFlows and Setup

2.1.1 Pre-delivered CPI iFlows

This section explains the pre-delivered CPI iFlow templates for creating a service ticket, purchase order, and sales order.

The following Cloud Platform Integration (CPI) iFlows are delivered as templates for integration with SAP IoT and are intended for reference only. As a customer, you must create your own iFlows based on your own system setup.

Post Service Notifications to C4C

This iFlow makes an OData service call to the SAP Cloud for Customer (C4C) to create a service request.

The iFlow is triggered from the action of a rules framework. This service accepts a payload from the action of a rules framework and calls an OData service in the C4C. The payload should be in a valid Service Notification Creation format (for example, the `ServiceRequestCollection`, `ServiceRequestDescriptionCollection`, or `ServiceRequestItemCollection` entity sets). This is a passthrough iFlow and the response from the C4C is passed directly back to the SAP IoT. SAP IoT consumes this to handle the downstream logging and processes. This iFlow service is called from an action configured in SAP IoT.

Partner Directory Config Parameters Used

PID	Parameter
`CNG`	ServicePost (Value should be `ServiceNotificationPost`)
`ServiceNotificationPost`	URL (C4C URL including entity set)
`ServiceNotificationPost`	Credential Name for ECC in HCI

Where Used	In Action configuration
Source Systems	Cloud for Customer (C4C)
Adapters	http sender/receiver

Post Sales Order to ECC

This iFlow makes an OData service call to the SAP ERP Central Component (SAP ECC) gateway system to create a sales order in a SAP ECC back-end system.

The iFlow is triggered from the action of a rules framework. This iFlow service accepts a payload from the action of a rules framework and calls an OData service in the SAP ECC gateway system. The incoming payload should be mapped to the OData service. The gateway OData service uses `BAPI_SALESORDER_CREATEFROMDAT2`.

This iFlow has the following prerequisites:

- Cloud Connector and RFC destination in SAP BTP must be configured.
- **The gateway service must be configured/developed in a gateway system connected to the back-end ECC system where sales orders are created.** This gateway service must map the inbound fields to the SAP ECC BAPI input structures and should map the outbound structures and return table.

This iFlow service is called from an action configured in SAP IoT.

Partner Directory Config Parameters Used

PID	Parameter
`CNG`	SalesOrderPost
`SalesOrderPost`	URL (Gateway service URL)
`SalesOrderPost`	Query
`SalesOrderPost`	Credential Name for SAP ECC in HANA Cloud Integration (HCI)

Where Used	In Action configuration
Calls	None
Source Systems	ECC (through a gateway service)
Adapters	http sender/receiver
Prerequisite	An OData service using `BAPI_SALESORDER_CREATEFROMDAT2` must exist in the gateway server connected to the ECC

Post Consignment Sales Order to ECC

This iFlow makes an RFC call using the CPI RFC sender adapter to create a Sales Order (SO) in an ECC back-end system. This iFlow is triggered from the action of a rules framework. The service accepts a payload from the action of a rules framework and calls `BAPI_SALESORDER_CREATEFROMDAT2` in the back-end ECC gateway system.

This is not strictly a consignment order creation iFlow. Depending on the value in the *Document Type* (DocType) field in the payload, different types of orders can be created, such as **`TA`** for a standard sales order or **`KB`** for consignment orders. This mapping step in the iFlow maps the inbound fields in the payload to the ECC BAPI

input structures. Depending on the business scenario, applicable fields from the payload should be mapped to the relevant ECC BAPI input structures to ensure that the back-end order has all the required attributes.

The `BAPIRETURN` table is returned from the back end to the sender. This iFlow service is called from an action configured in SAP IoT.

This iFlow has the following prerequisite:

- Cloud Connector and RFC destination in HCP must be configured.

Partner Directory Config Parameters Used

Externalized Parameter	Parameter
RFC Destination	{{Receiver_destination_0}}

Where Used	In Action configuration
Calls	None
Source Systems	ECC
Adapters	http sender/RFC receiver

Post Purchase Requisition to ECC

This iFlow makes an RFC call using the CPI RFC sender adapter to create a purchase requisition in an ECC back-end system. This iFlow is triggered from the action of a rules framework. The service accepts a payload from the action of a rules framework and calls `BAPI_REQUISITION_CREATE` in the back-end ECC gateway system.

This mapping step in the iFlow maps the inbound fields in the payload to the ECC BAPI input structures. Depending on the business scenario, the applicable fields from the payload can be mapped to the relevant ECC BAPI input structures to ensure that the back-end order has all the required attributes.

The `BAPIRETURN` table is returned from the back end to the sender. This iFlow service is called from an action configured in SAP IoT.

This iFlow has the following prerequisites:

- Cloud Connector and RFC destination in HCP must be configured.

Partner Directory Config Parameters Used

Externalized Parameter	Parameter
RFC Destination	{{Receiver_destination_0}}

Where Used	In Action configuration
Calls	None
Source Systems	ECC
Adapters	http sender/RFC receiver

Post Purchase Order to ECC

This iFlow makes an RFC call using the CPI RFC sender adapter to create a purchase order in an ECC back-end system. This iFlow is triggered from the action of a rules framework. The service accepts a payload from the action of a rules framework and calls `BAPI_PO_CREATE1` in the back-end ECC gateway system.

This mapping step in the iFlow maps the inbound fields in the payload to the ECC BAPI input structures. Depending on the business scenario, the applicable fields from the payload can be mapped to the relevant ECC BAPI input structures to ensure that the back-end order has all the required attributes.

The `BAPIRETURN` table is returned from the back end to the sender. This iFlow service is called from an action configured in SAP IoT.

This iFlow has the following prerequisites:

- Cloud Connector and RFC destination in HCP must be configured.

Partner Directory Config Parameters Used

Externalized Parameter	Parameter
RFC Destination	{{Receiver_destination_0}}
Where Used	In Action configuration
Calls	None
Source Systems	ECC
Adapters	http sender/RFC receiver

2.2 Workflow Integration

Backend integration can also help trigger both SAP BTP workflows and SAP S/4 workflows from SAP IoT. Workflows can be triggered in a similar way as iFlows. As a customer, you need to define the destination through the SAP BTP Destination service and provide the corresponding payloads.

To trigger a SAP BTP workflow, you need to define the destination as shown in the following sections.

Destination Type and Authentication Mechanisms

Authentication Mechanisms

Authentication	Description
Authentication	OAuth2ClientCredentials
Client ID	`<CLIENT ID>` of your SAP BTP Workflow Service (CF)
Client Secret	`<CLIENT SECRET>` of your SAP BTP Workflow Service (CF)

Authentication	Description
Token Service URL	`https://<SUB DOMAIN of your tenant>.authentication.<YOUR REGION>.hana.ondemand.com/oauth/token`
Token Service User	`<SAP BTP user>` to initiate the workflow (user needs to have the required permissions)
Token Service Password	`<SAP BTP password of above user>`

Destination Proxy Type: Internet

Destination URL: `https://api.workflow-sap.cfapps.<REGION>.hana.ondemand.com/workflow-service/rest/v1/workflow-instances`.

Example

A sample URL can be https://api.workflow-sap.cfapps.eu10.hana.ondemand.com/workflow-service/rest/v1/workflow-instances. For more information about SAP BTP Workflow, please refer to https://api.sap.com/api/SAP_CP_Workflow_CF/resource.

Sample Payload

Format: **JSON**

A sample payload (request body) has the following structure. Make sure that you replace the value of the `definitionId` with your own workflow `Id`, and remove the context variable placeholder.

Code Syntax

```
{
  "definitionId": "<WORKFLOW ID>",
  "context": {
<YOUR CONTEXT VARIABLES IN JSON FORMAT>
                      }
}
```

To trigger an *S/4 workflow*, you define the destination as below:

Destination Type and Authentication Mechanisms

Authentication Mechanisms

Authentication	Description
Authentication	BasicAuthentication
User	`<S/4 HANA user>` to initiate the workflow (You need to have the required permissions)
Password	`<S/4 HANA user password>`

Destination Proxy Type: OnPremise for On-Premise S/4 HANA Systems

> **i** Note
>
> This also requires the configuration of a Cloud Connector.

Destination URL: `http://<HOST NAME>:<PORT>/SAP/BC/WORKFLOW_XML/?~protocol=01&~localkey=<WORKFLOW ID>&sap-client=<CLIENT NUMBER>`.

Sample Payload

Format: **WF-XML**

A sample payload (request body) has the following structure. Make sure that you replace the value of the `<KEY>` with your own S/4 system and workflow `Id`.

> Code Syntax

```
<?xml version="1.0" ?>
<WfMessage Version="SAP.1.0" xmlns="http://www.wfmc.org/standards/doc/WF-XML">
    <WfMessageHeader>
        <Request>
            <ResponseRequired>Yes</ResponseRequired>
        </Request>
        <Key>https://mys4hana.sap.com:44300/SAP/BC/WORKFLOW_XML/?
~protocol=01&~localkey=WS02000061</Key>
        <Operation>CreateProcessInstance</Operation>
    </WfMessageHeader>
    <WfMessageBody>
        <CreateProcessInstance>
            <Key>https://mys4hana.sap.com:44300/SAP/BC/WORKFLOW_XML/?
~protocol=01&~localkey=WS02000061</Key>
            <ContextData>
                <YOUR_CONTEXT_VARIABLE 1></YOUR_CONTEXT_VARIABLE 1>
                <YOUR_CONTEXT_VARIABLE 2></YOUR_CONTEXT_VARIABLE 2>
            </ContextData>
```

```
            <StartImmediately>Yes</StartImmediately>
        </CreateProcessInstance>
    </WfMessageBody>
</WfMessage>
```

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 a 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.

www.sap.com/contactsap

© 2022 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.

THE BEST RUN SAP