PUBLIC

Document Version: 1.0 – 2021-04-01

SAP Workflow Service in the Cloud Foundry Environment

THE BEST RUN SAP

Content

1 Concepts. 5
1.1 Workflow Definition versus Workflow Instance. 5
1.2 Status Changes for Workflow Instances. 6
1.3 Status Changes for Task Instances. 7
1.4 Conventions, Restrictions, and Limits. 8
1.5 Supported Languages. 13
1.6 Browser Support. 15

2 Initial Setup. 17
Using the Automatic Setup. 17
Using the Manual Setup. 18
2.1 Assign Workflow Roles to Your Users. 19
2.2 Create a Dev Space. 20

3 Developing Applications with Workflow Service. 21
3.1 Modeling a Workflow. 21
Using SAP Business Application Studio. 22
Editor Layout. 25
Define Workflows. 26
Configure Custom Workflow Attributes. 71
Translate Workflows. 72
Transport Workflows between Accounts. 74
Build and Deploy the Workflow Module. 75
Accelerated Modeling with Speed Buttons. 76
Expressions. 78
Legacy: Enable the Workflow Editor in SAP Web IDE. 86
3.2 Creating User Interfaces. 106
Creating a Custom Task UI. 107
Creating a Custom Start UI. 120
Creating a Workflow Form. 132
3.3 Using Workflow APIs. 151
Determine Service Configuration Parameters. 152
Access Workflow APIs Using OAuth 2.0 Authentication (Authorization Code Grant). 154
Access Workflow APIs Using OAuth 2.0 Authentication (Client Credentials Grant). 156
Determine the Service Host. 157
Modifying the Context of a Workflow Instance. 157
Work with Attachments on a Workflow and Task Instance. 160

Updating Task Properties. . . . 163
Workflow Execution Log. . . . 166
Error Codes. . . . 167

4 Consuming the Workflow Service at Application Runtime. . . . 169
4.1 Configuring Workflow Service. . . . 169
Create a Service Instance of SAP Workflow Service Using the Cockpit. . . . 170
Create a Service Instance of SAP Workflow Service Using the Command Line Interface. . . . 171
Configure the Access to Workflow Data. . . . 173
Enable Technical Authentication. . . . 174
Configuring Principal Propagation for Service Tasks. . . . 176
Configure the Workflow Service Mail Destination. . . . 177
Configure Document Management for Workflow Service Attachments. . . . 180
Configure an SAP API Business Hub Destination. . . . 182
Create Workflow and My Inbox Tiles on Central Launchpad. . . . 184
Create Custom Start UI Tiles on the Central Launchpad . . . 211
Export SAP Workflow Service Data. . . . 213
Deactivate the Workflow Service. . . . 215
4.2 Using SAP Workflow Service. . . . 215
Working with Tasks in My Inbox. . . . 216

5 Security. . . . 227
5.1 Architecture. . . . 227
5.2 Identity Provider and Identity Management. . . . 229
5.3 Authorization Configuration. . . . 229
Technical Authentication. . . . 232
5.4 Destinations. . . . 232
Configure a Service Task Destination with OAuth2SAMLBearerAssertion for Principal Propagation. . . . 235
5.5 Data Protection and Data Privacy. . . . 237
Information Report. . . . 237
Erasure. . . . 237
Change Log. . . . 238
Glossary. . . . 239
5.6 Audit Logs. . . . 240

6 Monitoring and Troubleshooting. . . . 243
6.1 Managing Workflows Using the Monitor Workflows App. . . . 243
Managing Workflow Instances. . . . 244
Managing Task Instances. . . . 246
Managing Workflow Definitions. . . . 248
Deep Linking in Monitor Workflows App. . . . 250
6.2 Traceability during Transport of Workflow Modules. . . . 251

6.3 Verify the Availability of Workflow Extensions in SAP Business Application Studio. 252

1 Concepts

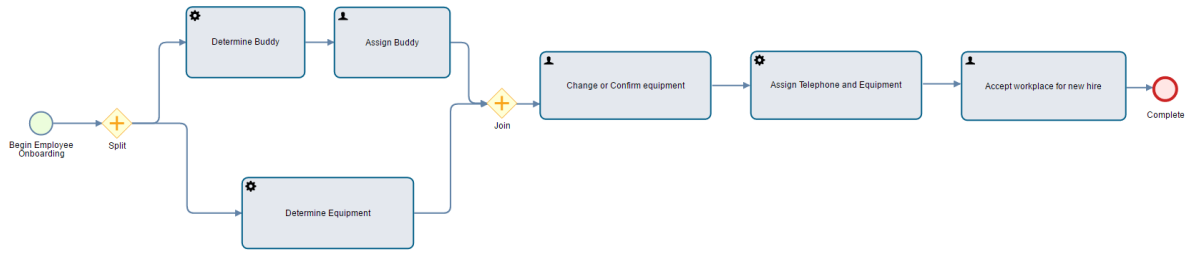
The SAP Workflow service offers modern process automation capabilities.

Related Information

Business Process Model and Notation

1.1 Workflow Definition versus Workflow Instance

A workflow is a collection of linked automatic or human activities that serve a certain goal.



Example Workflow Depicted as a Diagram

The workflow service differentiates between workflow definitions and workflow instances. A workflow definition specifies:

- Which actions should be performed
- When these actions should be performed
- The circumstances under which these actions should be performed

The actual execution of these actions is called a workflow instance. So, a single workflow definition can have multiple workflow instances. This differentiation is essential for monitoring and troubleshooting. You can also define a subject for a workflow that helps business users track these instances.

These different notions of "workflow"are both used in the workflow service. In the context of design time, "workflow relates to a workflow definition. In the runtime context, "workflow" refers to a workflow instance.

The same holds true for tasks. In the context of design time, "task" refers to the specification of a certain type of activity. Whereas a runtime task relates to a particular activity to be performed instantiated from the corresponding specification.

> **i Note**
>
> Do not confuse "workflow instance" as described here in the workflow context with "workflow service instance". The latter refers to the SAP BTP, Cloud Foundry environment instance concept.

1.2 Status Changes for Workflow Instances

Workflow instances follow a status and action model.

A started workflow instance moves into the RUNNING status and that means:

- All execution branches can be executed.
- The workflow engine processes the next workflow elements unless the branch is asynchronously waiting for external activation. This activation can be a user task completion, a message event, or a timer event.

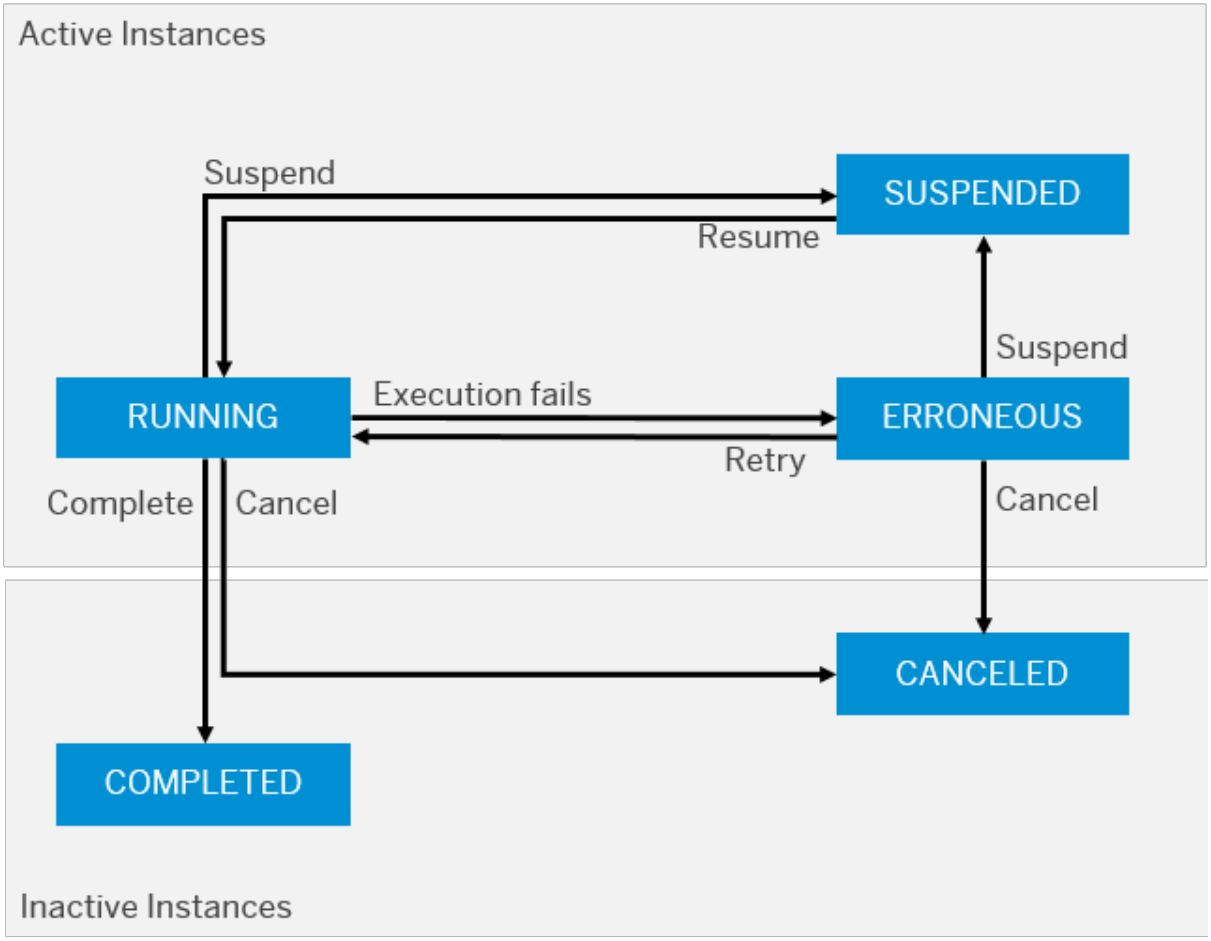
If the execution of a workflow element fails, it is retried several times. After that, the workflow element is kept but is not executed again. The workflow instance then changes to the ERRONEOUS status. However, only the execution branches with failed workflow element executions are affected. Parallel branches without failures continue to execute. For erroneous instances, you can reset the execution counter manually with the retry action.

You can move an instance that cannot reach any end event, or is no longer required, to the CANCELED status. You can also temporarily move an instance to the SUSPENDED status and resume it later.

An instance that reaches at least one terminating end event, or all non-terminating events, is moved to the COMPLETED status.

Start status of an instance: RUNNING

Final status of an instance: CANCELED, COMPLETED



The status and action model for workflow instances is exposed for customer consumption in the REST API. For more information, see Workflow Instance Status.

1.3 Status Changes for Task Instances

User tasks follow a status and action model, which is reflected in the REST API.

When a new user task is created without a processor its status is READY.

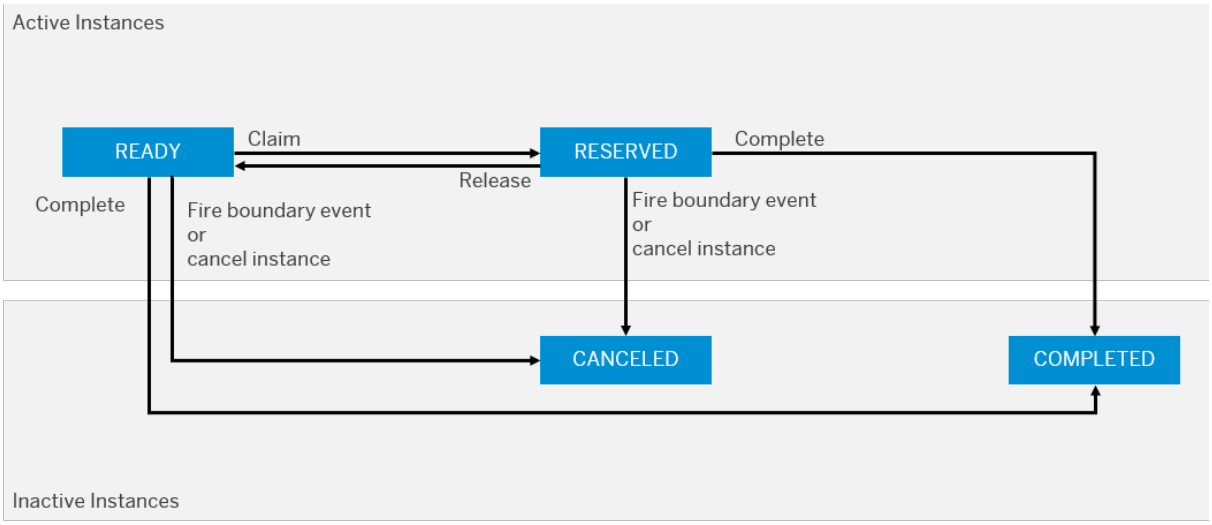
When a recipient claims the task, its status changes to RESERVED. When the user releases the task again, its status reverts back to READY.

With the REST API you can complete a task and set its status to COMPLETED.

A user task has the status CANCELED when a canceling boundary event on the user task triggers it or when the workflow instance of the task is canceled. For more information, see Configure Boundary Timer Events [page 31] and Managing Workflows Using the Monitor Workflows App [page 243].

Start status of an instance: READY

Final status of an instance: CANCELED, COMPLETED



1.4 Conventions, Restrictions, and Limits

These conventions, restrictions, and limits apply to the workflow service.

Considering this information during development, helps you to achieve an optimal use of the service.

> **i Note**
>
> Limits are, to the extent possible, subject to change.

Execution Limits

Area	Limit	Value for Standard Plan (Paid Account)	Value for Lite Plan (Trial Account)	More Information
Workflow context	Size of the workflow context	100 KB per workflow service instance		• Creating and Reading Workflow Context Structures [page 52] • Applies also if exceeded only temporarily • Applies to any operation on the workflow context, that is, to all types of tasks and all types of APIs.

Area	Limit	Value for Standard Plan (Paid Account)	Value for Lite Plan (Trial Account)	More Information
API	Request rate limit	150 requests per second and tenant	20 requests per second and tenant	• Using Workflow APIs [page 151] • Includes requests triggered from user interfaces delivered by SAP. • In exceptional situations, requests are temporarily rate-limited to a lower value than the given value.
	Request body size	512,000 bytes		n.a.
	Processing time	30 seconds		Includes response generation by the server
Script tasks	Execution time	150 milliseconds		Configure Script Tasks [page 51]
Service tasks	Connection timeout	1 minute		Time to establish the connection with the remote host
	Socket timeout	3 minutes		Maximum period between two data packets
	Total execution time	4 minutes		For recommendations on how to implement service tasks if high execution times are common, see Configure Service Tasks [page 45].
Deployments	Number of deployments per tenant	-	80	Counting the number of versions of all workflow definitions.
Workflow instances	Number of workflow instances	-	200	All workflow instances in a tenant, regardless of the status.

Area	Limit	Value for Standard Plan (Paid Account)	Value for Lite Plan (Trial Account)	More Information
Workflow activities within a workflow instance	Number of workflow activities that can be executed before the system interrupts further execution	2000	150	After executing this number of activities, for example, tasks, events, or gateways within a workflow instance, the instance is set to the ERRONEOUS status and preliminarily stopped. The limit applies when this happens for the first time in the instance. The workflow instance can be retried within certain limits, see below.
	Number of retries available after the system had interrupted the execution	10	3	The number of times a workflow instance can be retried after its execution was stopped because the number of activities in this instance had exceeded the limit. Each retry adds a number of additional executable activities, see below.
	Number of additional activities available after retrying an instance that was interrupted	200	15	A workflow instance can execute this number of activities, before the system interrupts again and sets the instance to the ERRONEOUS status again. If all retries and the respective number of additional activities are used up, further additional activities are only available through SAP support.

Restrictions

- UI5 Version
  - To use My Inbox, you need UI5 version 1.71 (latest patch) or higher.
  - To render workflow forms (see Creating a Workflow Form [page 132]) you need UI5 version 1.71 (latest patch) or higher.
- Workflow Forms
  - A form can have at most 100 fields or 100 sections at root level.
  - A section can have at most 100 fields or 100 subsections.
  - A subsection can have at most 100 fields.
  - A field with a dropdown or a radio buttons control can have at most 100 selectable values.
  - For task forms, at most 100 attachments are displayed.

- Variable Names
  There are many ways to create, change, or delete variables in the context of a process. For example, when starting the process, using script tasks, updating the process context manually. In all cases, the names of the variables in the process context must adhere to the following rules:
  - Must not start with "SAP_WFS".
  - Must start with a letter (latin alphabet, or A-Z, a-z).
  - Can contain additional letters, digits, and underscores.
- Duration
  When expressions are used to specify duration, they must resolve to ISO 8601 format during runtime. For more information, see ISO 8601 .
  However, you must consider the following:
  - The smallest units, which the duration specification supports are minutes.
  - The "Week" unit ("W") isn't supported.
  - The duration specification supports integers only. Also, while using the static mode, the value of the duration field must be less than 2147483647.
- Workflow Definition ID
  The workflow service instances that are directly created in a subaccount and those created while subscribing to an application that uses the workflow service, share the content in the same subaccount. Hence, the Workflow Definition ID must be unique across all service instances and all subscriptions. This is checked while deploying a workflow module.

  > → Recommendation
  >
  > We recommend that you use namespaces to avoid workflow definition ID clashes.
  >
  > For example, **`com.sap.bpm.workflows.LeaveRequest`** or **`com_sap_bpm_workflows_LeaveRequest`**.

- Service Tasks
  - Custom Headers
    - The following headers are not supported and you can't deploy them:
      - Authorization
      - Connection
      - Proxy-Authorization
      - Proxy-Connection
      - TE
      - Trailer
      - Transfer-Encoding
      - Upgrade
      - Content-Length
      - HTTP2-Settings
      - Host
      - SAP-PASSPORT
      - SAP-CLIENT
    - The maximal size of all custom headers must not exceed 8 KB.
  - Content Negotiation (media types for `Accept/Content-Type`)

Custom headers allow content negotiation in general, however, the request and response data is mapped from/to the workflow context. As the workflow context is based on JSON, we strongly recommend that you rely on related media types only.
As a precaution, the workflow service evaluates the content type of a response and only parses content that reportedly is JSON. If the response contains a different media type or the actual content is not JSON, processing fails.

- Header Precedence
  There are cases, in which the headers you define in the properties of the service task are overridden. This can happen, for example, if you configure the CSRF path or the destination settings so that additional headers become part of the request.

Model Limits

The following model limits apply to the workflow service.

Common Properties

Property Name	Limit
Name	64 characters
Documentation	2000 characters

Workflow Properties

Property Name	Limit
Subject	255 characters
Business Key	255 characters
Custom Workflow Attribute	15 custom workflow attributes per workflow definition at a time. 30 unique attributes per custom workflow attributes across all workflow versions. Currently, only type string is supported. The ID and the label of an attribute can be 255 characters long. The definition of the value in the model can't exceed 4000 characters. Also, after expression evaluation at runtime, the value of an attribute can't exceed 4000 characters.

Flow Element Properties

Flow Element	Property Name	Limit
Intermediate Message Event	Message Name	128 characters
	Response Variable	255 characters
User Task	Subject	255 characters

Flow Element	Property Name	Limit
	Description	2000 characters
	Users	Maximum of 100 users, maximum of 255 characters per user
	Groups	Maximum of 100 users, maximum of 255 characters per user
	Custom Attribute	15 custom attributes per user task at a time, 30 unique attributes per user task across all workflow versions Currently, only type string is supported. The ID and the label of an attribute can be 255 characters long. The definition of the value in the model can't exceed 4000 characters. Also, after expression evaluation at runtime, the value of an attribute can't exceed 4000 characters.
Service Task	Destination	200 characters
	Path	7000 characters
	Path to XSRF Token	7000 characters
	Request Variable	255 characters
	Response Variable	255 characters
Script Task	Script File	10000 characters
Mail Task	To, Cc, Bcc	Maximum of 100 e-mail addresses that can contain a maximum of 5000 characters
	Subject	1000 characters
	Mail Body (Plain or HTML)	10000 characters

1.5 Supported Languages

The workflow service is available in the following languages.

- Workflow documentation:
  - Chinese (Simplified)
  - English
  - Japanese

- Workflow API (error responses): English
- Workflow editor: English
- Monitoring user interfaces for workflow administrators:
  - Arabic
  - Chinese (Simplified)
  - Czech
  - Danish
  - Dutch
  - English (UK)
  - English (US)
  - French
  - German
  - Hebrew
  - Hungarian
  - Italian
  - Korean
  - Japanese
  - Norwegian
  - Polish
  - Portuguese (Brazil)
  - Russian
  - Spanish (Mexiko)
  - Spanish (Spain)
  - Turkish
- Inbox application for workflow participants:
  - Arabic
  - Bulgarian
  - Catalan
  - Chinese
  - Chinese trad.
  - Croatian
  - Czech
  - Danish
  - Dutch
  - English
  - Estonian
  - Finnish
  - French
  - German
  - Greek
  - Hebrew
  - Hindi
  - Hungarian
  - Italian
  - Japanese

- Kazakh
- Korean
- Latvian
- Lithuanian
- Malay
- Norwegian
- Polish
- Portuguese
- Romanian
- Russian
- Serbian (Latin)
- Slovak
- Slovenian
- Spanish
- Swedish
- Thai
- Turkish
- Ukrainian
- Vietnamese

To translate SAP Fiori launchpad entities, see Translate Your Site.

Related Information

Translate Workflows [page 72]

1.6 Browser Support

For the UIs of the workflow service, the following browsers are supported on Microsoft Windows PCs and where mentioned on Mac OS X.

> **i** Note
>
> The workflow editor doesn't support the Safari browser.

Supported Browsers

Browser	Versions
Mozilla Firefox	Extended Support Release (ESR) and latest version
Google Chrome	Latest version

Browser	Versions
Safari	7.0 and upwards (for Mac OS X only)

For a detailed list of SAPUI5 supported browsers and platforms, see Browser and Platform Support - SAPUI5.

2 Initial Setup

Some preparatory steps are needed before you can use the SAP Workflow service.

Prerequisites

- You've either a global or a trial account that is entitled to use the workflow service.
  For more information, see Get a Global Account or Get a Trial Account.
- When using a global account, the following additional prerequisites need to be fulfilled:
  - You're a global account administrator.
  - You've created a space within a subaccount in which SAP BTP, Cloud Foundry environment is enabled.
    For more information, see Managing Orgs and Spaces Using the Cockpit and Account Administration in the Cloud Foundry Command Line Interface.
  - Within that space, your user is assigned to the *Space Developer* role for the subaccount. You need this role to execute the configuration steps.
    For more information, see Managing Authorization in Global Accounts and Subaccounts [Feature Set B] and Assign Role Collections.
    For more information, see Subaccount Member Roles and Roles.
- For SAP BTP, Cloud Foundry environment trial, you directly access the SAP Business Application Studio from the https://cockpit.hanatrial.ondemand.com/cockpit/#/home/trial page by choosing *SAP Business Application Studio*.

Using the Automatic Setup

You set up the workflow service for your global account of type enterprise or trial using a booster.

Procedure

1. Access your global account page in the SAP BTP cockpit, and choose the *Boosters* from the left-hand navigation.
2. On the *Set up account for Workflow Management* tile, choose *Start*.
3. Follow the wizard to select your subaccount, organization, and space for the initial setup.

   The booster sets up the cockpit for all SAP Workflow Management capabilities: Workflow, Business Rules, Process Visibility, and Workflow Management. Currently, it sets up the following:

   - Assigns entitlements and quota for: SAP Workflow service, SAP Business Rules service, SAP Process Visibility service, and SAP Workflow Management.

> **i Note**
>
> SAP Workflow service is added with the service plan *standard*.

- Creates a space, if not already available.
- Enables subscriptions for SAP Business Application Studio and the Workflow Management.
- Creates a service instance for each of these: workflow service, business rules service, process visibility service, workflow management service.
- Creates the following destinations: BUSINESSRULES_APIHUB, BUSINESS_RULES, WM_CF_SPACE_PROVIDER, and SAP_API_Business_Hub.
- Assigns all role collections of the workflow management service to the user.

You can also use the booster of workflow management service to set up your trial account. There, the booster uses the service plan *lite*.

4. For some use cases, for example, to create a custom SAP Fiori launchpad tile, you have to add further entitlements to your subaccount.
   a. From the SAP BTP cockpit, navigate to your global account and access the subaccount.
   b. From the navigation area, choose *Entitlements*, and choose *Configure Entitlements*.
   c. Choose *Add Service Plans*, search and select the entitlement needed, and choose *Add Service Plans*.

Related Information

Configuring Workflow Service [page 169]
Trial Tutorial: Set up your account using Booster

Using the Manual Setup

You can set up the workflow service manually, although we recommend using the booster setup.

Procedure

1. From the SAP BTP cockpit, navigate to your global account and assign the entitlement for the service plan *standard* of the workflow service to your subaccount.

   For more information, see Configure Entitlements and Quotas for Subaccounts.

   For your trial account, use the booster of Workflow Management. The booster uses the service plan *lite*.
2. To use SAP Business Application Studio with workflow extensions, do the following:
   a. Subscribe to the SAP Business Application Studio. For more information, see Subscribe to Multitenant Applications Using the Cockpit.

      This enables the *Go to Application* link to access the application. You can then share the link with your developers.

b. Assign the developers a role collection that contains the special role to access the application. For more information, see Manage Authorizations.
c. Create a dev space in SAP Business Application Studio where you build your project and workflow module.

   For more information, see Create a Dev Space [page 20].

To use SAP Web IDE, from the SAP BTP, Neo environment account and enable the workflow feature if not already enabled by default. To be able to use the SAP Web IDE, you must have an SAP Business Technology Platform (SAP BTP) SAP BTP, Neo environment subaccount. For more information, see Creating Your Neo Subaccount

3. To enable developers to develop applications with the workflow service, assign the `WorkflowDeveloper` role. For more information, see Authorization Configuration [page 229].

Related Information

Assign Workflow Roles to Your Users [page 19]
Configuring Workflow Service [page 169]
Configure Entitlements and Quotas for Subaccounts

2.1 Assign Workflow Roles to Your Users

To assign roles to users, you need to add roles to one or more role collections and then assign these role collections to your users.

Prerequisites

You are assigned to the *User & Role Administrator* role in the *Security* section of your subaccount.

Procedure

1. Navigate to your subaccount in the SAP BTP cockpit.
2. In the navigation area, under *Security*, choose *Role Collections*.
3. Choose the + icon *Create New Role Collection*.
4. Enter a name and optionally a description, then choose *Save*.
5. Select the new role collection, then choose *Edit*.
6. Under *Role Name*, select the role from the dropdown list.
7. Choose *Save*.

8. In this way, keep adding roles or create additional role collections according to your requirements.
9. To assign the role collection to a group of users, enter the group name in the *Users Groups* section, under *Name*.
10. Choose *Assign Role Collection*, then select the desired role collection and choose *Assign Role Collection*.
11. Choose *Save*.

Related Information

Authorization Configuration [page 229]

2.2 Create a Dev Space

You create your workspace where you build the workflow.

Procedure

1. In your web browser, open the SAP BTP cockpit.
2. Choose *SAP Business Application Studio*.
3. In SAP Business Application Studio, choose *Create Dev Space*.
4. Enter a name, for example, `mydevspace`, and make sure to select *SAP Fiori* or *SAP Cloud Business Application* and in addition the *Workflow Management* extensions.
5. Choose *Create Dev Space* again.

   Wait until the dev space is created and you see the RUNNING status.
6. Click the *Dev Space* to access the workspace.
7. If you have never accessed the workspace before, open it with the *Open Workspace* button.
8. Select the *projects* folder and choose *Open*.

   The *PROJECTS* explorer is opened.

Related Information

Create a Basic Workflow in SAP Business Application Studio (Tutorial)

3 Developing Applications with Workflow Service

Developer tasks for the SAP Workflow service service that are executed in the workflow editor or in the workflow runtime.

Related Information

Concepts [page 5]
Modeling a Workflow [page 21]
Creating User Interfaces [page 106]
Build and Deploy the Workflow Module [page 75]
Using Workflow APIs [page 151]

3.1 Modeling a Workflow

You can model a workflow using the workflow editor in SAP Business Application Studio or in SAP Web IDE Full-Stack. In the SAP BTP, Cloud Foundry environment, the recommended IDE is SAP Business Application Studio.

> **i** Note
>
> The workflow editor does not support Safari browser.

Modeling a workflow includes the following steps, which you can perform using the workflow editor in SAP Business Application Studio or in SAP Web IDE Full-Stack:

- Defining a start point of the workflow: Define a start point of the workflow using the start event. For more information, see Events [page 64].
- Defining workflow steps and their sequence: Define the process steps using the following graphical objects:
  - Tasks: There are user tasks that are performed by a human or a mail program, service or script tasks that are performed by the system. For more information, see Tasks [page 28].
  - Gateways: Gateways control the flow of execution in a workflow. For more information, see Gateways [page 68].
- Defining an endpoint of the process: Defines an endpoint of the process using end event or terminate end event. For more information, see Events [page 64].

Related Information

Legacy: Enable the Workflow Editor in SAP Web IDE [page 86]
Model Workflows in a Multitarget Application Project with SAP Web IDE [page 86]
Build and Deploy the Workflow Module [page 75]
Accelerated Modeling with Speed Buttons [page 76]
Expressions [page 78]
Migrate Workflow Project to a Multitarget Application Project with SAP Web IDE [page 104]
SAP Business Application Studio - Developer Guide

3.1.1 Using SAP Business Application Studio

Before you can start to model workflows, you need to execute some preparatory steps to use SAP Business Application Studio with workflow extensions.

Procedure

1. Subscribe to the SAP Business Application Studio. For more information, see Subscribe to Multitenant Applications in the Cloud Foundry Environment Using the Cockpit.

   This enables the *Go to Application* link to access the application. You can then share the link with your developers.
2. Assign the developers a role collection that contains the special role to access the application. For more information, see Manage Authorizations.

   > **i Note**
   >
   > The application identifier (providing the role) varies in different landscapes and isn't necessarily the same as described in the linked documentation.

3.1.1.1 Generate an MTA Project Containing a Workflow Module with SAP Business Application Studio

You need to create a multitarget application (MTA) project that contains a workflow module.

Prerequisites

In SAP Business Application Studio, you've created a dev space with the following extensions:

- SAP Predefined Extension *MTA Tools* that comes, for example, with the application kinds *SAP Fiori* or *SAP Cloud Business Application*
- Additional SAP Extension *Workflow Management* that you need to select

Procedure

1. Open a Dev Space to model workflows as part of a multitarget application.
2. Inside your dev space open a new workspace, choose *Menu* > *File* > *Open Workspace*.
3. Select your folder of choice, for example, projects and choose *Open*.

   > **i Note**
   >
   > The SAP Business Application Studio might reload completely.

4. Open a terminal window to start the Yeoman generator, choose *Menu* > *Terminal* > *New Terminal*.

   A new terminal window opens.
5. Navigate to your workspace folder, for example, `cd /home/user/projects`.
6. Type in **`yo`**, and choose ENTER to start the generator framework.
7. Select *Basic Multitarget Application* and enter the required details, for example:

   ```
   ? Enter a project name my-project.
   ```
8. Navigate inside the project by entering the following command:

   ```
   cd my-project
   ```
9. Type in **`yo`**, and choose ENTER to start the generator framework.
10. Select @workflow/workflow Module and fill the required details, for example:

    ```
    ? Module Name workflow

    ? Workflow Name MyWorkflow

    ? Workflow Description
    ```

    > **i Note**
    >
    > - The deployment of a workflow module checks the uniqueness of a workflow definition ID across all service instances and subscriptions. See Conventions, Restrictions, and Limits [page 8].

- Select "overwrite" to create a new `mta.yaml` file which contains the already existing mta configurations, merged with the newly added workflow module.

11. Typically, you do not need to change anything more.

    The plan is set depending on your environment:
    - Trial: lite
    - Production: standard
    - To use an existing instance of workflow service, define it by using `org.cloudfoundry.existing-service` resource type along with the existing resource name.

      You can find the existing instance name of workflow service by navigating to *Services* > *Service Instances* in the cockpit. See the code snippet for an example.

      Sample Code

      ```
      resources:
        - name: <existing_workflow_service_instance_name>
          type: org.cloudfoundry.existing-service
      ```

    For more information about resources types and parameters, see MTA Module Types, Resource Types, and Parameters for Applications in the Cloud Foundry Environment.

Results

The workflow editor opens and you can begin modeling your workflow.

3.1.1.2 Create a Workflow with SAP Business Application Studio

Create a workflow from scratch.

Procedure

1. Select the project in the *Explorer* view of SAP Business Application Studio.
2. Choose *View* > *Find Command* > *Workflow: Create New Workflow*.
3. Enter the required details and confirm.

   The workflow editor opens.

   Note

   The deployment of a workflow module does not check the uniqueness of a workflow definition ID across all service instances and subscriptions. See Conventions, Restrictions, and Limits [page 8].

Therefore, we recommended that you specify a namespace for your workflow. The namespace is generated automatically into the workflow ID.

3.1.1.3 Open Workflow Files in the Workflow Editor with SAP Business Application Studio

Open existing workflow files in the workflow editor to view or modify them.

Procedure

1. Log on to SAP Business Application Studio.
2. In the *Explorer* view, open your Dev Space and workspace.
3. Select the multitarget application (MTA) project, and navigate to the workflow module.
4. Navigate to the *workflows* folder and open the file.

   This starts the workflow editor.

> **i Note**
>
> SAP Business Application Studio lazily loads the extensions from a central repository. Depending on your network connection it could be that the editor is not (yet) ready. Instead a text editor is shown. Wait for some time, then close the editor and try to open it again. Should the issue persist, see Verify the Availability of Workflow Extensions in SAP Business Application Studio [page 252].

3.1.2 Editor Layout

The workflow editor consists of the following areas:

- Canvas: The canvas renders and models the workflow, which connects flow objects such as events, tasks, and gateways.
- Palette: The palette contains flow objects, for example, events, tasks, and gateways. You can easily model your workflow by selecting the required flow object in the palette and placing it on the canvas using click and drop.
- Toolbar: The toolbar contains tools such as undo, redo, delete and auto layout options.
- Properties: The properties view provides configuration options for flow objects.
- Diagram Overview: When a workflow model is bigger than the canvas layout, diagram overview can help you visualize where the current view is in the diagram. Also, you can navigate to the required part of the workflow.

3.1.3 Define Workflows

You use this procedure to define workflows.

Procedure

1. Open the workflow with the *Workflow Editor*.

   For more information, see Open Workflow Files in the Workflow Editor with SAP Business Application Studio [page 25].
2. In the *Subject* field of *Workflow Properties* pane, provide the text that helps you identify the workflow instances started for this workflow definition.

   > **Example**
   >
   > For the employee onboarding process, you can consider a *Subject* like "`Employee onboarding process initiated for ${context.employeename}`". For more information, see Expressions [page 78].

   > **Note**
   >
   > - For more information on character limits for workflow service, see Conventions, Restrictions, and Limits [page 8].
   > - The data accessed in expressions must be available at the latest when the expression is resolved, for example, when creating a user task. If the data isn't available, some parts of the expression, or the whole expression won't be resolved. For more information, see Workflow Definition versus Workflow Instance [page 5].
   > - A workflow definition ID is generated for every workflow that you model. This ID is used when you start a new workflow instance. For more information, see the Workflow Instances section in Using Workflow APIs [page 151].
3. In the *Business Key* field under the *General* tab of the *Workflow Properties* pane, provide an optional identifier for workflow instances based on business data.

   The business key can include static text as well as expressions similar to the workflow subject. With the business key, you can later identify a workflow instance without knowing the technical instance ID.

> ### Example
>
> For the employee onboarding process, you can consider a business key based on the unique employee ID, for example, "${context.employeeid}". With this you can, for example, search for a specific workflow instance using the employee ID instead of the technical workflow instance ID.

> ### Note
>
> In SAP Workflow service uniqueness isn't enforced for business keys neither globally nor within a specific workflow definition. If you require a one-to-one relationship between a business key value and a workflow instance, make sure that you use business data within your business key expression that uniquely identifies the entities processed within the workflow. You can, for example, use the order ID or the employee ID.

4. Navigate to the *Attributes* tab under *Workflow Properties* pane to add the attributes for a workflow. To configure these attributes, see Configure Custom Workflow Attributes [page 71].
5. To model the start event of a workflow, select ○ ▶ *Events* > *Start Event* and drop it onto the canvas from the palette.
6. In the *Start Event Properties* pane, provide a name and documentation for the start event.

> ### Note
>
> A unique *ID* gets generated for every workflow artifact. This *ID* is in read-only mode.

7. (Optional) Configure a sample context while modeling a start event. After the deployment of the workflow, the sample context is displayed in the *Monitor Workflows* app while starting a new workflow instance. For more information, see Configure Start Events [page 65].

   You can also retrieve the configured sample context using the Public API.

   For more information, see SAP Workflow service API
8. To add a task to the workflow, see Tasks [page 28].
9. To add a gateway to the workflow, see Gateways [page 68].
10. To add an intermediate message event, see Configure Intermediate Message Events [page 66].
11. To add an intermediate timer event, see Configure Intermediate Timer Events [page 67].
12. To connect two flow elements, choose the  icon.

> ### Note
>
> If you choose a flow element using the speed button, the connection automatically appears. In this case, the above step isn't required. To connect two flow elements, choose the  icon, keep the mouse button pressed on the required flow element and move your cursor to the next flow element that needs to be connected in the workflow.

   For more information about speed buttons, see Accelerated Modeling with Speed Buttons [page 76].
13. To model the end event of a workflow, choose ◎ ▶ *Events* > *End Event* and drop it onto the canvas from the palette.
14. In the *End Event Properties* pane from the first flow pane, provide a name and documentation for the end event.

15. To model the end of a workflow as a terminate end event, choose *Events* > *Terminate End Event* and drop it onto the canvas from the palette.

    For more information on the terminate end event, see Events [page 64].

    > **i Note**
    >
    > You can also model a terminate end event using the speed buttons. For more information, see Accelerated Modeling with Speed Buttons [page 76].

16. In the *Terminate End Event Properties Properties* pane, provide a name and description for terminate end event.
17. To format the workflow model, choose *Arrange Horizontally* or *Arrange Vertically* from the toolbar.
18. Choose *Save*.

    > **→ Recommendation**
    >
    > - We recommend that you save the changes before exiting. If you don't, your changes are lost.
    > - Each time you change the properties of flow elements, make sure that you press the ENTER key.

3.1.3.1 Tasks

SAP Workflow service editor supports the following tasks:

- *User Task*: A flow object that illustrates a task that a human performs. User tasks appear in My Inbox where the processor of the task can complete the task instance, and view its description.
- *Service Task*: A flow object that illustrates a system task, for example, calling an external service. A service task is performed immediately, when the process execution arrives at it.
- *Script Task*: A flow object that illustrates a script that gets executed when the process execution arrives at it. This is an automated activity.
- *Mail Task*: A flow object that you configure to send e-mails to one or more recipients.

Related Information

Configure User Tasks [page 29]
Configure Service Tasks [page 45]
Configure Script Tasks [page 51]
Configure Mail Tasks [page 60]

3.1.3.1.1 Configure User Tasks

You must use this procedure when you want a user to perform a particular task in the workflow.

Context

You can propagate the user who completes a task to a service called later by a service task in the same workflow instance. For more information, see Configure Service Tasks [page 45].

Procedure

1. Choose  (Tasks), then *User Task* from the palette and drop it on to the canvas.
2. Select the user task icon that you dropped on the canvas.
3. In the *User Task Properties* area, choose the *General* tab.
4. Provide a *Name* and *Documentation* for the user task.

   > **i Note**
   >
   > - For more information on character limits for workflow service, see Conventions, Restrictions, and Limits [page 8].
   > - A unique *ID* gets generated for every workflow artifact. This ID is in read-only mode.
   > - Ensure that the Name field is short, precise, and contains a sufficiently unique identifier, as it's displayed to the end users. For example, in My Inbox.

5. From *User Task Properties* area, choose the *Details* tab.
6. Depending on the priority of the user task, choose one of the following options from the *Priority* menu:
   - Low
   - Medium (default)
   - High
   - Very High
7. In the *Display Texts* section, provide the following details:
   - *Subject*: Title of the task instance.
   - *Description*: Any additional information.
8. In the *Recipients* section, name the users who should process the task. Either indicate individual users or groups of users in the *Users* or *Groups* field. Refer to the *Subject Name Identifier* setting of your identity provider to see which user record attribute to use, for example, email address or logon names.

   > **i Note**
   >
   > - *Subject*, *Description*, *Users*, and *Groups* can also refer to the dynamic workflow context. For example, if you want to provide a *Subject* that references a variable from dynamic context, you can specify the expression in *Subject* field as "`Approval for ${context.employee.name}`". For more information, see Expressions [page 78].

For users and groups, either use a context reference that resolves to a string with different users or groups separated by commas or use a context reference that resolves to an array of strings.
- To provide multiple users or groups of users to process the task, separate each unique ID with a comma.
- You can assign a maximum number of 100 users or groups as recipients to a user task.
- To have a role collection as a recipient in the *Recipients* section, provide the desired role collection in *Groups*.
- Recipients can view these tasks in My Inbox. They can also complete these tasks, which further proceed the workflow execution.

9. To configure the duration by when the task is due (date relative to the task creation time), select the *Configure Due Date* checkbox.

   You must configure the due date using the following substeps:

   a. To provide a duration for the due date as an expression, choose *Expression* from the *Due Date Based On* dropdown. Now, provide the due date in the *Duration* field as an expression.

   > **i Note**
   >
   > You must provide an expression in the *Duration* field using a subset of the ISO 8601 format. For example, `PT${context.minutes}M`. The JUEL expression `${context.minutes}` is evaluated at runtime. You can provide multiple duration attributes by using multiple JUEL expressions. For more information about the duration formats that are supported in ISO 8601, see Conventions, Restrictions, and Limits [page 8].

   b. To provide a duration for the due date as a static value, choose *Static Value* from the *Due Date Based On* dropdown. Now, provide the due date in the *Duration* field as a numeric value, and choose a *Unit of Time*.

10. To display information about the task execution in the inbox workflow log, select *Show in inbox workflow log*.
11. To allow forwarding of the task to another end user in the My Inbox app or using the Inbox API, see the Inbox API for Cloud Foundry or Inbox API for Neo, select *Allow Forward*.

    > **i Note**
    >
    > - A forwarded user does not need to be a configured recipient. However, this user becomes one of the task recipients. That means, the task remains accessible to this user even after it has been released.
    > - The passed user ID is not validated. If the given user does not exist, an administrator is required to reassign the task.
    > - Make sure that the My Inbox tile configuration contains the `userSearch` parameter and is set to false, see Create Workflow and My Inbox Tiles on Central Launchpad [page 184] and Legacy: Create Workflow and My Inbox Tiles on SAP Fiori Launchpad [page 192].

12. Configure a custom task user interface.

    You have the following options:

    - Configure a Custom Task User Interface Using an HTML5 App [page 33]
    - Configure a User Task UI Using Workflow Forms [page 38]

13. Assign custom attributes to a user task. For more information, see Configure Custom Task Attributes [page 40].

14. (Optional) To include a timer for the user task, add a boundary timer event. For more information, see Configure Boundary Timer Events [page 31].
15. Connect the user task to the required flow elements.
16. Choose *Save*.

Related Information

Accelerated Modeling with Speed Buttons [page 76]
Configure a Custom Task User Interface Using an HTML5 App [page 33]
Configure Boundary Timer Events [page 31]
Conventions, Restrictions, and Limits [page 8]

3.1.3.1.1.1 Configure Boundary Timer Events

Configure a boundary timer event to trigger an alternative flow if a user task doesn't finish within a specified time duration.

Context

Boundary timer events are attached to a user task. Some user tasks may need to be completed during a certain time interval. You can add a boundary timer event to define the duration of time for which the flow can wait at the user task before starting an alternative flow. There are two types of boundary timer events:

- Canceling Boundary Event: When this event is triggered, it cancels the user task it is attached to.
- Non-Canceling Boundary Event: When this activity is triggered, it does not cancel the user task it is attached to.

> **Example**
>
> In an employee onboarding scenario, the equipment assignment to a new hire must be confirmed by the buddy assigned to the new hire. The buddy is responsible for confirming the equipment that needs to be procured for the new hire.
>
> A non-canceling boundary timer event can be modeled on the "Confirm or Change Equipment" user task to send a reminder mail to the buddy if the task is not completed in three days. Similarly, a canceling boundary timer event can be modeled where the duration is such that the timer elapses two days before the joining date of new hire. Additionally, an alternative escalation flow, such as an escalation email, must be sent to the manager of the buddy to take required action; in this case, the original "Confirm or Change Equipment" task becomes irrelevant. Hence, the 'Confirm or Change Equipment' user task is canceled.

Procedure

1. Choose *Boundary Timer* from the speed button of the required user task.
2. Provide a *Name* and *Documentation* for the boundary timer event.
3. In the *Boundary Timer Event Properties* area, choose the *Details* tab.
4. Provide the waiting duration for the flow in the *Duration* (date field relative to the task creation time). You can use one of the following ways to configure this field:
   - To use expressions, choose *Expression* from the *Duration Based On* dropdown.

     > **i Note**
     >
     > You must provide an expression in the *Duration* field using a subset of the ISO 8601. For example, `PT${context.minutes}M`. The JUEL expression `${context.minutes}` is evaluated at runtime. Consider specifying an expression that evaluates to a string containing only digits. This avoids ambiguous data type conversions. You can provide multiple duration attributes by using multiple JUEL expressions. For more information about the duration formats that are supported in ISO 8601, see Conventions, Restrictions, and Limits [page 8].

   - To use a static value, choose *Static Value* from the *Duration Based On* dropdown. Now, provide the *Duration* as a numeric value, and choose a *Unit of Time*.
   - To use the due date value as the duration, choose *Task Due Date* from the *Duration Based On* dropdown.

     > **i Note**
     >
     > Duration for the boundary timer event is set to the due date value provided in the respective user task.

5. To define the boundary timer event as canceling, select the *Cancel Task* checkbox.
6. Choose *Save*.

> **i Note**
>
> - You can add multiple boundary timer events to a user task, which gets triggered when the corresponding timers are fired. When a canceling boundary event is triggered, any boundary events attached to the same task that haven't yet triggered are canceled.
> - One specific case needs to be taken into account: namely, suspending and resuming a workflow instance with several boundary timer events on an active user task. If such an instance is resumed and it has been suspended for a time period longer than the corresponding timer durations, there is no deterministic order in which the events are triggered.
> - When you add multiple boundary timer events, they are placed on the same position at the bottom of the user task. This may lead to several events on top of each other. However, these events can be moved along the boundary of the user task.

3.1.3.1.1.2 Configure a Custom Task User Interface Using an HTML5 App

With SAP Workflow service, end users can access their workflow tasks in their inboxes with user interfaces.

Context

HTML5 applications are supported for user tasks. For information about how to create such applications, see Creating a Custom Task UI [page 107]. To use an HTML5 application for user tasks, it needs to be configured.

Procedure

1. To embed a custom task UI for HTML5 applications that are available in the workspace, perform the following:
   a. Choose the *User Interface* tab.
   b. In the *HTML5 App Name* section, select the *SAPUI5 component* type. In the *HTML5 App Name* section, choose *Select*.
   c. In the *Choose User Interface* window, choose the *Project Name* from the list of projects that are available in the workspace.

      > **i Note**
      >
      > - Based on the selected *Project Name*, an *Application Name* is predicted. You can also provide a different application name by editing this field.
      > - *Application Name* is the name of the deployed application on SAP Business Technology Platform (SAP BTP).
      > - Application names containing a dash character („-“) are currently not supported.

   d. Choose *SAPUI5 Component Path* from the dropdown menu.
   e. Choose *OK*.

      > **i Note**
      >
      > *SAPUI5 Component* is added automatically but can be edited. If the selected HTML5 application is configured for a managed approuter, then also the *Business Solution Name* is added automatically. That entry is editable as well.

2. To manually provide the custom task UI details, provide the following details in the *User Interface* tab:

Property Name	Sample Value	Description
HTML5 App Name	employeeonboarding	Name of the deployed HTML5 application on SAP Business Technology Platform (SAP BTP).
SAPUI5 Component	sap.demo.Waas	SAPUI5 component name without the `<.component>` suffix
Business Solution Name	workflowtaskuis	This property is only applicable and mandatory for HTML5 applications configured for an "Approuter Managed by SAP Business Technology Platform (SAP BTP)". The business solution name refers to the service, as defined in the manifest JSON file of the HTML5 application. **i Note** The value is must be written without any dots.

3. In the *Parameters* field, provide the configuration data that can be accessed at runtime.

   For example, `key1=value1,key2=${context.value2}`

   If the same HTML application needs to be used for different task UIs with minor modifications, the URL parameter can be used to define the modification.

   For example, a task UI contains three actions namely: Accept, Reject, Rework. If the Rework action is not required for some specific tasks, you can still reuse this UI. This can be done by passing some parameters as configuration data. The task UI developer can then access these parameters in the custom task UI and choose to show or hide specific actions. For more information, see Access the User Task Data [page 114].

   i Note

   - You can enter multiple key value pairs separated by comma. If the value contains a comma, then you can use a backslash as shown below:
     `key1=value1,key2=value2\,value3\,value4`
     The key values in this case are as follows:
     key1=value1
     key2=value2, value3, value4
   - The following points must be considered while providing keys:
     - Keys cannot contain JUEL expressions and must be static.
     - Keys must start with a letter and can contain only alphanumeric characters.
     - Keys cannot contain whitespaces or special characters except for underscores.
   - Values can be static or can contain JUEL expressions. For more information, see Expressions [page 78].

- Expressions are evaluated at task creation time, that is, they cannot be used to transfer data to the user interface that changes after this time. For such cases, see Set the Task and Task Context Models [page 110].

4. Save the workflow.

3.1.3.1.1.3 Configure a Custom Task User Interface Using an HTML5 App with SAP Web IDE Full-Stack

Context

> **i Note**
>
> This procedure only applies to existing subaccounts that use SAP Fiori launchpad modules. If you have a subaccount that was created after January 15, 2021, see Configure a Custom Task User Interface Using an HTML5 App [page 33].

Procedure

1. To embed a custom task UI for projects that are available in the workspace, perform the following:
   a. Choose the *User Interface* tab.
   b. In the *HTML5 App Name* section, select the *SAPUI5 component* type. Under *HTML5 App Name* section, choose *Select*.
   c. In the *Choose User Interface* window, choose the *Project Name* from the list of projects that are available in the workspace.

      > **i Note**
      >
      > - Based on the selected *Project Name*, an *Application Name* is predicted. You can also provide a different application name by editing this field.
      > - *Application Name* is the name of the deployed application on SAP Business Technology Platform (SAP BTP).

   d. Choose *SAPUI5 Component Path* from the dropdown menu.
   e. Choose *OK*.

      > **i Note**
      >
      > *SAPUI5 Component* is added automatically, which is editable.

2. To manually provide the custom task UI details, provide following details in the *User Interface* tab:

Property Name	Description
HTML5 App Name	Name of the HTML5 application
Component URL	Location of `<Component.js>` in the HTML5 project
SAPUI5 Component	SAPUI5 component name without the `<.component>` suffix

Configuration of the above *User Interface* properties varies based on the following scenarios:

- Grunt build is not enabled in SAP Web IDE Full-Stack or `SAPUI5 Client Build` is not enabled on SAP Web IDE.
  Open the component.js file of the UI5 application. The sample screenshot is given and the corresponding *User Interface* properties are shown in the following table:

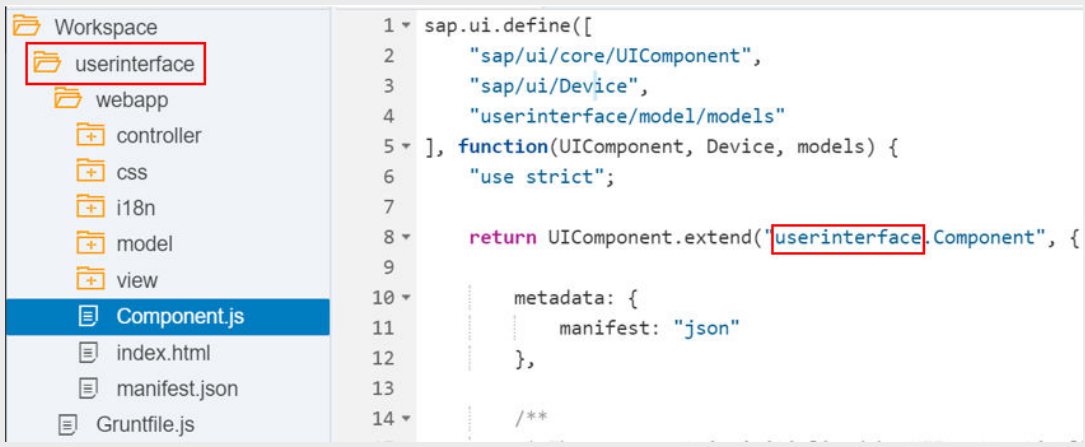
  Example

Property Name	Value
HTML5 App Name	`employeeonboarding`
Component URL	`webapp`
SAPUI5 Component	`sap.demo.Waas`

- Grunt build is enabled in SAP Web IDE Full-Stack or `SAPUI5 Client Build` is enabled on SAP Web IDE. For more information, see Building Applications Using Grunt and Building Applications Using SAPUI5 Build.
  Configuration of the above *User Interface* properties based on this scenario is illustrated with an example:

Example



Property Name	Value
HTML5 App Name	<UI5 project name>
Component URL	
SAPUI5 Component	<UI name>

Note

Component URL in this case is the location of the *Component.js* file, relative to the webapp folder.

3. In the *Parameters* field, provide the configuration data that can be accessed at runtime.

   For example, `key1=value1,key2=${context.value2}`

   If the same UI5 component needs to be used for different task UIs with minor modifications, the URL parameter can be used to define the modification.

   For example, a task UI contains three actions namely: Accept, Reject, Rework. If Rework action is not required for some specific tasks, you can still reuse this UI. This can be done by passing some parameters as configuration data. The task UI developer can then access these parameters in his/her custom task UI and choose to show or hide specific actions. For more information, see Access the User Task Data [page 114].

   ### Note

   - You can enter multiple key value pairs separated by comma. If the value contains a comma, then you can use backslash as shown below:
     `key1=value1,key2=value2\,value3\,value4`
     The key values in this case are as follows:
     key1=value1
     key2=value2, value3, value4
   - The following points must be considered while providing keys:
     - Keys cannot contain JUEL expressions and must be static.

- Keys must start with a letter and can contain only alphanumeric characters.
- Keys cannot contain whitespaces or special characters with an exception of underscores.

- Values can be static or can contain JUEL expressions. For more information, see Expressions [page 78].
- Expressions are evaluated at task creation time, that is, they cannot be used to transfer data to the user interface that changes after this time. For such cases, refer to Set the Task and Task Context Models [page 110].

4. Save the workflow.

Related Information

Configure User Tasks [page 29]

3.1.3.1.1.4 Configure a User Task UI Using Workflow Forms

With SAP Workflow service, end users can access their workflow tasks in their inboxes with user interfaces.

Context

Workflow forms are supported for user tasks. For information about how to create a task form, see Creating a Workflow Form [page 132]. To use a task form for user tasks, it needs to be configured.

Procedure

1. In the workflow editor, select the user task and choose *User Interface*.
2. Under *Type*, choose *Form*.
3. Under *Form Details*, choose one of the following options:
   - *Create File*
     On the *New Form* dialog, enter the following data:

Field	Description
Name	Name of the form you create
ID	Identifier of the new form

Field	Description
Revision	Revision of the form For more information, see Versioning Forms [page 147].

> **i Note**
>
> In user tasks, you can only refer to task forms. Therefore, the type *Task Form* is preset and cannot be changed when you create a form in the workflow editor.

- *Select*
  On the *Select Form* dialog, enter the following data:

Field	Description
Workflow Module	Name of the workflow module. The name is preset to the current workflow module. You cannot change the name.
File Name	Name of the form from the list of forms that are available in the current module only
Form Revision	Revision of the form For more information, see Versioning Forms [page 147].

For more information on forms, see Creating a Workflow Form [page 132].

4. Save the workflow.

   The form is created in a separate *forms* folder within the workflow module in a folder named exactly like the workflow for which the form is created.

Related Information

Configure User Tasks [page 29]
Creating a Workflow Form [page 132]

3.1.3.1.1.5 Configure Custom Task Attributes

You can assign custom task attributes to user tasks.

Context

With custom task attributes, you can define business-related properties and assign them to user tasks, such as project ID or project name.

At runtime, you can use the respective workflow service API or Inbox API to search for custom task attributes or to find the respective task instances. For more information about the characteristics of the various APIs, see Using Workflow APIs [page 151].

Procedure

1. Choose the *Attributes* tab.
2. To add a row, choose *Add*.

   > **i Note**
   >
   > You can reorder the added custom attributes by using *Move Up* or *Move Down*.

3. Provide the following details in the table:

Name	Description
ID	A unique identifier of the attribute within a user task. **i Note** You can only use alphanumeric characters as ID and it must only start with an alphabet.
Label	A human readable name of the attribute, which appropriate user interfaces can use to label the attribute.
Type	Data type of the attribute. **i Note** Currently, only the data type string is supported.

Name	Description
Value	A constant or an expression, which gets resolved upon task creation. For JUEL expressions, only the `${context.xyz}` subset is supported. `${info}`, `${roles}`, and `${usertasks}` aren't supported. A complete array isn't resolved in JUEL expressions. For `${context.aArray}`, for example, values of that kind of expression always resolve to null. However, for a single array element it resolves, for example, `${context.aArray[0]}`. A complex object is also not resolved in JUEL expressions. For `${context.complexObject}`, for example, values of that kind of expression always resolve to null. However, for a single object element it resolves. For example, `${context.complexObject.primitiveProperty}` resolves to the value of the primitive property. ${context} doesn't contain any node. It's also not supported and its value always resolves to null.

> **i Note**
>
> Labels as well as the order, in which the corresponding APIs return the task attributes, are taken from the latest versions of the workflow definition where these attributes are present.
>
> A user task can contain up to 15 attributes at a time. For more information, see Conventions, Restrictions, and Limits [page 8].

4. Save the changes.

Related Information

Configure User Tasks [page 29]
Display Custom Attributes in My Inbox [page 41]

3.1.3.1.1.5.1 Display Custom Attributes in My Inbox

You can display business-related data that is defined to workflow tasks, also known as *custom attributes*, in My Inbox. For example, the project ID or project name. This additional workflow contextual data can help business users work more efficiently with My Inbox.

For more information, see Custom Attributes in My Inbox [page 223].

> **i Note**
>
> Please, note that by default this feature is disabled in My Inbox. To enable it, the administrator has to configure the additional parameter **`showAdditionalAttributes=true`** in the app configuration of My Inbox.

Context

My Inbox supports two types of custom attributes:

1. custom attributes with SAP reserved IDs;
2. custom attributes with IDs defined by administrator.

The custom attributes of the above types are visualized by My Inbox app.

Use

1. Custom Attributes with SAP Reserved IDs

The custom attributes with SAP reserved IDs are *CustomTaskTitle*, *CustomNumberValue*, *CustomNumberUnitValue*, *CustomObjectAttributeValue*, and *CustomCreatedBy*.

They allow you to:

- Customize the task title
- Customize the *Created By* information
- Display a number or unit value
- Display additional information about a task

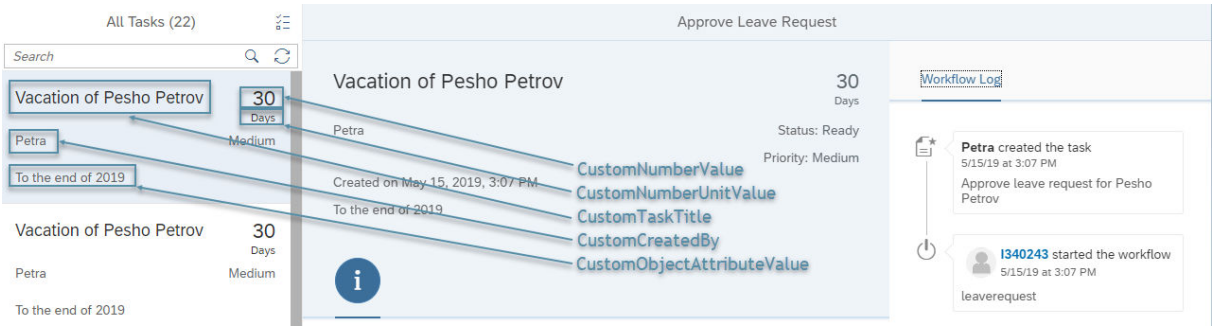
> **i Note**
>
> Please note that sorting or filtering is not supported by these values. In addition, they are not visualized as columns in the *Expert View* of My Inbox.

The below table aims to provide guidance when to use the custom attributes with SAP reserved IDs in the *Details View* of the standard task UI for task details, and information about their visualization in the My Inbox UI.

		Visualization		
Custom Attribute ID	**Usage**	**Master-Detail List**	**Expert View**	**Generic UI for Task Details**
CustomTaskTitle	Replaces the title of a task with a custom value	☑	☑	☑
CustomNumberValue	Displays a KPI indicator in the Details View of the standard task UI for task details	☑		☑
CustomNumberUnitValue	Displays a KPI unit	☑		☑

Custom Attribute ID	Usage	Visualization		
		Master-Detail List	Expert View	Generic UI for Task Details
CustomObjectAttributeValue	Displays additional contextual information about the task	☑		☑
CustomCreatedBy	Replaces the name of the task creator with a custom value	☑	☑	☑

The screenshot below visualizes where the custom attributes with SAP Reserved IDs are visualized in the *Master-Detail View*.



Procedure

- Use one of the following SAP Reserved IDs as ID for the custom attribute, depending on the result you would like to achieve. See the above table.
  - *CustomTaskTitle*
  - *CustomNumberValue*
  - *CustomNumberUnitValue*
  - *CustomObjectAttributeValue*
  - *CustomCreatedBy*

To assign custom attributes to your tasks, see Configure Custom Task Attributes [page 40].

2. Other Custom Attributes with IDs Defined by Administrator

These custom attributes are custom attributes with different from the SAP Reserved IDs, and defined by you, as an administrator. They are displayed in the generic UI for task details of *Master-Detail View* and the *Expert View* ofMy Inbox. In addition, they can be exposed as columns in the *Expert View* of My Inbox.

They allow you to:

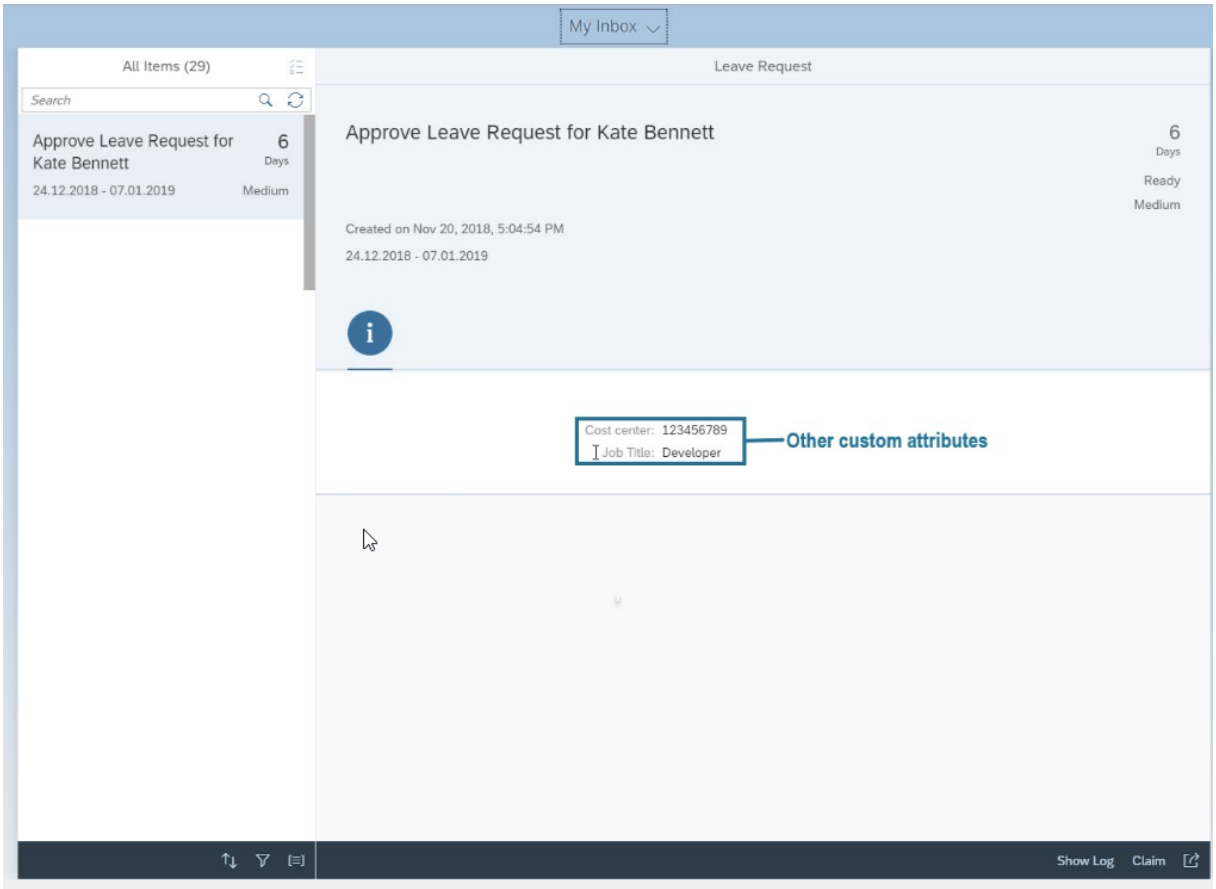
- Expose additional information about the task in the Information tab of the generic UI for task details in My Inbox.
- Expose the custom attributes as columns in the *Expert View* of My Inbox. See Expert View [page 219].
- Enable business users to quickly find tasks by sorting and filtering based on custom attributes in the *Expert View*.

Procedure

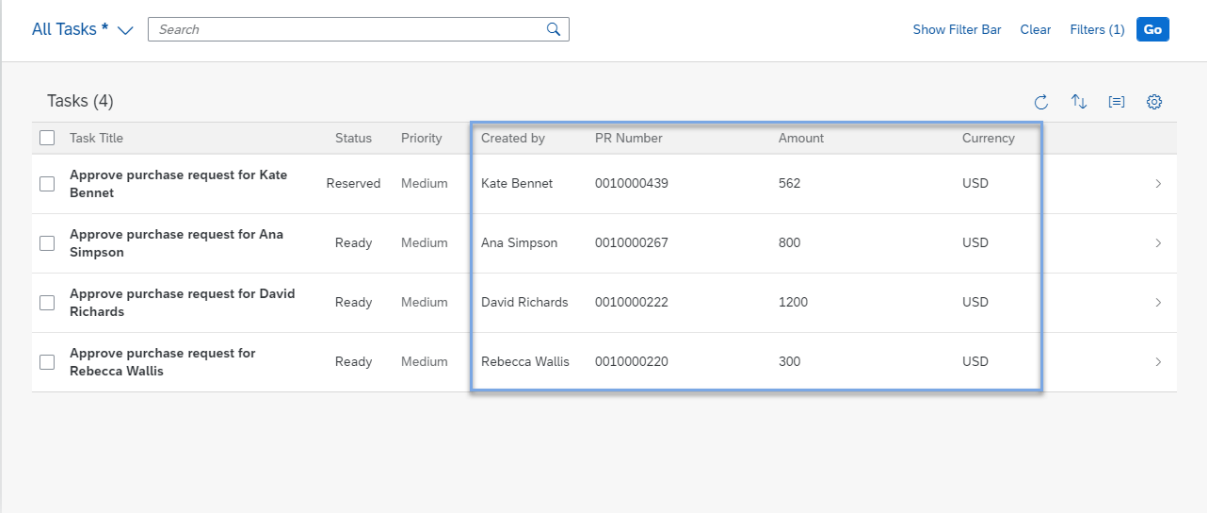
- As ID for the custom attribute, assign a random ID, which is different from the SAP Reserved IDs for custom attributes.

As a result, the additional information displayed by custom attributes is visualized within the *Master-Detail View* of My Inbox, in *Information tab* under *Task Description*.

> **i Note**
>
> Please, note that sorting and filtering are not possible based on this custom attribute data in the *Master-Detail View*. The *Expert View* can be used instead.



The custom attributes are visualized as columns in the *Expert View* of My Inbox.



To assign custom attributes with IDs defined by administrator to your tasks, see Configure Custom Task Attributes [page 40].

> **i Note**
>
> In case the tasks are using a custom task UI, the custom attributes will not be visualized as part of the custom task UI.

> **i Note**
>
> You can define up to 15 other custom attributes per task, which are displayed in the information tab of the default task UI of My Inbox (if no custom UI is configured). If you use a custom task UI, these custom attributes are not displayed. In this case, at runtime, you can use the respective Workflow ServiceAPI or Task Consumption ModelAPI to search for custom attributes or to find the respective task instances. For more information about the APIs, see Using Workflow APIs [page 151].

3.1.3.1.2 Configure Service Tasks

If you want the system to perform a particular task in the workflow, configure a service task.

Context

The execution of service tasks is subject to resource limits, for example, with respect to network timeouts. If the target service doesn't comply with the time restrictions described in Conventions, Restrictions, and Limits [page 8], the connection with the target service is aborted, the service task fails, and the workflow instance is put into the ERRONEOUS state.

Long execution times negatively impact the execution of other tasks of a specific tenant, because there's only a limited number of parallel executions allowed for a tenant. The resource limits enforced by the workflow service therefore have the purpose of freeing up resources as early as possible for other tasks.

→ Tip

We recommend that the service execution time is much less than the limits documented in Conventions, Restrictions, and Limits [page 8]. If high execution times are common, consider building an intermediate service that initiates asynchronous processing of the actual service call and returns quickly. It can report back the execution result using message events. For more information, see Configure Intermediate Message Events [page 66].

Workflow service executes automatic retries when a service task fails. A service task execution is only considered successful, if the HTTP status code is in the range 200 to 299. Ensure that these retries can complete successfully, for example, after administrative intervention and even in case of communication failures. Such cases may occur, for example, when the client doesn't receive the information about the actual server-side success of the call. If the called services aren't appropriately implemented, that is, they aren't idempotent, the retries from the workflow service might fail permanently or create duplicate entities.

Certain workflow service REST APIs aren't idempotent and shouldn't be called to modify the currently running workflow instance. For more information and recommendations, see 2884301.

Procedure

1. Choose  (Tasks), then *Service Task* from the palette and drop it on to the canvas.
2. Select the service task icon that you dropped on the canvas.
3. In the *Service Task Properties* area, choose the *General* tab.
4. Provide a *Name* and, optionally, a description in the *Documentation* field for the service task.

   A unique read-only *ID* is generated for every workflow artifact.
5. In the *Service Task Properties* area, choose the *Details* tab.
6. Provide the *Destination*.
   - The destination you provide here is the destination specified in the consumer subaccount, which determines the host to connect to at runtime. For more information about the supported feature set of destinations, see Destinations [page 232].
   - Destination can also refer to the dynamic workflow context. For more information, see Expressions [page 78].
   - For more information on character limits for SAP Workflow service, see Conventions, Restrictions, and Limits [page 8].
7. Select one of the following options from the *Choose a Service From* list:
   - *SAP API Business Hub*
   - *Others* (default)

   SAP API Business Hub is the central catalog, hosted by SAP to discover, explore, and test the SAP and partner APIs that are required to build extensions, or process integrations using SAP Business Technology Platform (SAP BTP). For more information, see SAP API Business Hub.
8. If you have chosen *SAP API Business Hub*, perform the following procedure Configure a Service from SAP API Business Hub [page 50].
9. If you selected *Others*, provide the following details:

a. *Path*: Resource path that appends to the URL of the specified destination while calling the service.
   - Path can consist of variables. For more information, see the below example.
   - Services that are called from a service task must support the JSON format for request and response body. Consequently, the workflow service sends the `<Content-Type: application/json>` header in every HTTP request, and expects the service to return `<Accept: application/json>`. Other responses are declined by the workflow service runtime, which can lead to a runtime error.
   - Ensure the URL that is concatenated from the *Destination* and the *Path* are valid.
   - The workflow service runtime ensures proper encoding of the final URL that is invoked. To avoid a double encoding, don't enter the URL specified at the destination, the value for the path property, and `xsrf` path property in an encoded format.
b. *HTTP Method*: Specify one of the following HTTP methods: GET, POST, PATCH, PUT, or DELETE.

   If the HTTP method is POST, DELETE, PATCH, or PUT, then the *Path to XSRF Token* field appears. XSRF token is used for modifying operations that are protected against XSRF (cross-site request forgery) attacks. For more information, see SAP Business Technology Platform (SAP BTP).
c. *Path to XSRF Token*: The resource path that must be appended to a specified destination, while calling the service to fetch an XSRF token. If the authentication type of the specified destination is OAuth related, you probably don't need to enter a value here.
d. *Request Variable*: Link to a workflow context node that populates the body of the HTTP request.
   - The referenced node is used 1:1 as content for the request body.
   - The complete context can be referenced in the request body as follows: `${context}`.
   - If the request variable contains a primitive JSON type (number or string) or literal (null, true, or false), the service must accept an HTTP body following RFC 8259 instead of the older RFC 4627.
   - If the HTTP method is POST, PATCH, or PUT, then you see the *Path to XSRF Token* field.
e. *Response Variable*: Link to a workflow context node that is created or overwritten to finally store the body of the HTTP response.
   - REST services can return successful status codes, for example, `204 No content`, without a response body. In such cases, the response variable reference is overwritten with a null value, if provided. Note that many services, especially OData services, respond with a 204 status code on successful PUT and PATCH operations.
   - The referenced node stores the response body.
   - The complete context can't be overwritten by the contents of the response body. As a result, the expression `${context}` can't be used in the response variable. A variable within the context must be specified to be used as a response variable. For example, `${context.leaveRequest}` or `${context.leaveRequest.response}` are valid response variables.

> **Example**
>
> This example shows how to call a REST service to store employee's leave requests. This service is XSRF protected.
>
> The service URL for this example is `https://{host}/leaverequest`.
>
> - *Destination* is created in the SAP Business Technology Platform (SAP BTP) subaccount with the following URL: `http://<host>:<port>`.
> - *Path*: `/leaverequest`
> - *HTTP Method*: *POST*
> - *Path to XSRF Token*: `/leaverequest/v1/xsrf-token`

- *Request variable*: `${context.leaveRequest.request}`
- *Response Variable*: `${context.leaveRequest.response}`

This code represents the sample payload.

Sample Code

```
leaverequest
{
"request":
{
"employeeId":"000001",
"startDate": "2016-10-10T00:00:00.000Z",
"endDate": "2016-10-19T00:00:00.000Z",
"reason":"vacation"
}
}
```

At runtime, context is added with the response variable when the service task is invoked. Once the service task is invoked, the context is appended with the response variable and looks like:

Sample Code

```
leaverequest
{
"request":
{
"employeeId":"000001",
"startDate": "2016-10-10T00:00:00.000Z",
"endDate": "2016-10-19T00:00:00.000Z",
"reason":"vacation"
}
"response":
{
status: "Successfully stored"
}
}
```

10. Enable principal propagation.
    a. Select *Principal Propagation* for the service task.

       For more information, see Configuring Principal Propagation for Service Tasks [page 176].
    b. In the *Flow Element* section, choose *Select*.

       This field is available only if principal propagation is active. Then, it's a mandatory field.
    c. To search for the start event or a user task in the workflow, use *Select Flow Element*.
       - To propagate the user who started the workflow instance, browse for the start event in the same workflow model.
       - To propagate the user who completed a user task instance, browse for the user task in the same workflow model.
       - If a user task is located in a loop, the last completion action of a corresponding task instance in a workflow instance defines the actual user that is propagated.
    d. Choose *OK*.
11. Model header parameters.

    For more information, see Model Header Parameters in the Workflow Editor [page 49].

12. Connect the service task to the required flow elements.
13. Choose *Save*.

Next Steps

In the SAP BTP, Cloud Foundry environment, you can deploy the workflow model into the workflow service runtime if you have the `SpaceDeveloper` role in the target account.

To make the workflow operational, an administrator must create and configure the destination mentioned by the workflow developer. For more information, see Destinations [page 232].

Related Information

Accelerated Modeling with Speed Buttons [page 76]
Connectivity Options Blog
Consume SAP Gateway OData Service in SAP Workflow Service

3.1.3.1.2.1 Model Header Parameters in the Workflow Editor

Context

> **i Note**
>
> There are cases, in which the headers you define in the properties of the service task are overridden. This can happen, for example, if you configure the CSRF path or the destination settings so that additional headers become part of the request. See the restrictions for header parameters in Restrictions [page 10].

Procedure

1. In the *Properties* section of the service task, choose the *Header* tab.
2. To add a row, choose *Add*.

   You can reorder the added header parameters by using *Move Up* or *Move Down*.

   To remove a row, first select it, then choose *Delete*.
3. Provide the following details:

1. Name (required) - The HTTP header name.
   You can only use alphanumeric characters, plus the following special characters: !, #, $, %, &, ', *, +, -, ., ^, _, `, | ~
2. Value - A constant, a JUEL expression, or a mixture of constants and JUEL expressions. It is resolved before issuing the HTTP request.
   You can use any printable ASCII characters, including the space. If a JUEL expression resolves to an invalid value, the service task execution fails. In case the JUEL expression can't be resolved, the unresolved JUEL expression is passed.

4. Save your changes.

3.1.3.1.2.2 Configure a Service from SAP API Business Hub

You use this procedure to use an API from SAP API Business Hub in the service task properties.

Prerequisites

When using SAP Business Application Studio, SAP API Business Hub is set as a destination. See Configure an SAP API Business Hub Destination [page 182].

Procedure

1. In the *Service* section, choose *Select* to browse for a required API.
2. In the *Select API from Business Hub* popup, select the required API from the table.
3. Choose *Next*.
4. Choose a resource from the *Resources* section.
5. Based on the resource selected, choose a method from the table.
6. Choose *Next*.

> **i Note**
>
> If you choose the method type as POST, PATCH, or PUT, then the *Request Variable* field appears. This field is auto populated, but it can also be edited.

7. From the *Response Variable* field, you can modify or keep the auto populated response name.

> **i Note**
>
> The *Request Parameters* and the *Response* sections are read-only.

8. Choose *Finish*.

> **i Note**
>
> HTTP method type, path, request/response variables are populated based on the selection. The *Path to XSRF Token* field is auto populated, if the APIs pushed to SAP API Business Hub have the `x-sap-csrf-token-path` attribute configured.

3.1.3.1.3 Configure Script Tasks

A script task is an automatic activity. When a workflow execution arrives at the script task, the corresponding script is executed.

Context

> **→ Recommendation**
>
> We recommend that you export and import workflow projects, rather than individual workflows, as additional script resources are added to the workflow project.

Procedure

1. Choose  (Tasks), then *Script Task* from the palette and drop it on to the canvas.
2. In the *Script Task Properties* area, provide a name and documentation (optional) for the script task.

   > **i Note**
   >
   > A unique read-only *ID* is automatically generated for every workflow artifact.

3. To add JavaScript files, perform one of the following steps:
   - Choose *Select* to browse for the JavaScript file in the current project.
   - To create a JavaScript file, perform the following steps:
     1. Choose the *Create File* link.
     2. In the *Create New File* window, provide a file name.
     3. Choose *Create*.
     4. In the JavaScript file, provide the script.

   > **i Note**
   >
   > - You can view and or edit the JavaScript file by selecting the *Script File* link.
   > - You can find the JavaScript file in the following location: `<workflow-module>/scripts/<workflow-name>/<script-file-name>.js`.
   > - For more information about *Code Editor*, see Developing Applications.

- The provided APIs, as well as the objects and arrays stored in the workflow context, are non-native JavaScript objects; that is, ECMAScript host objects. Their behavior might differ from that of the native objects. For more information about supported APIs, see:
  - Creating and Reading Workflow Context Structures [page 52]
  - Accessing Contextual Information During Execution of Script Tasks [page 56]
- The script must be in JavaScript that is based on ECMAScript 5.1. For more information, see the Ecma Web page. Restrictions: `'eval'` and `'Function'` are not supported for script tasks. Using the `function` keyword is supported, but you cannot assign functions to workflow context variables.
- The execution of script tasks is subject to resource limits, for example, with respect to processing time or memory usage. The limits enforced by the workflow service have the purpose of freeing up resources as early as possible for other tasks. The limits protect against excessive usage, for example, caused by in-efficient programming or unexpected input sizes. If the limits are exceeded, the corresponding workflow instance is put into the ERRONEOUS state. The error is written to the error logs of the workflow instance. You can retrieve the error logs using the REST API. If your scripts reach the resource limits, analyze the reasons, for example, large input data. Try to reduce the input size or the complexity of the transformations executed on it.
  For the specific limits that apply to script tasks, see Conventions, Restrictions, and Limits [page 8].

4. Save the workflow.

Related Information

Accelerated Modeling with Speed Buttons [page 76]
Conventions, Restrictions, and Limits [page 8]

3.1.3.1.3.1 Creating and Reading Workflow Context Structures

You can insert scripts to use library functions to manipulate the workflow context.

To interact with the workflow context, use the predefined identifier `'$.context'`. Data that is stored in the workflow context, for example, during the workflow start or from a previous script task, can be read, modified, or enhanced using a dot-notation as shown in the examples below. Such data might consist of either primitive data types that are supported by JavaScript (for example, a string or numeric value), or complex structures (for example, objects or arrays).

In general, the workflow context can only contain data that can also be represented using the JavaScript Object Notation (JSON). That is, the workflow context cannot store:

- Functions
- Prototype objects
- Special numbers, such as NaN (Not a Number), positive infinity, or negative infinity

> **i Note**
>
> In general, do not store large objects in the workflow context, but only the keys to more appropriate storages. See the “Claim Check” integration pattern. For data privacy reasons, we recommend deleting data, especially personal data, as soon as it is no longer needed.

Context changes are committed at the end of the script execution. Therefore, if the execution of the script task runs into an error, data that has been modified before within the same script task is not visible to subsequent activities in the workflow. This section describes how to interact with primitive variables in the workflow context. For complex structures, see *Related Information*.

Reading Variables

```
// variables are accessible as properties of $.context
var myAlias = $.context.myString;
// reading a not-existing variable returns null
if ($.context.myVariable === undefined)
{
// initialize myVariable lazy
}
```

```
// iterate over the variable names
for (var key in $.context) {
// use key, for example, $.context[key] to retrieve the variable value
}
// iterate over the object keys
for (var key in $.context.myObj) {
// use key, for example, $.context.myObj[key] to retrieve the object property
value
}
```

Setting Variables

```
// variables have to be assigned to $.context to be persisted
$.context.myString = myValue;
// variable assignments can also be chained
$.context.newString = $.context.newString2 = "new field value";
// variables of primitive type have a "copy-by-value" behavior
// $.context.myString will keep the value 'hello' after the following code
var myNewValue = "hello";
$.context.myString = myNewValue;
myNewvalue = "goodbye";
// myString will not be accessible in later steps of the workflow,
// if set as follows (but only as a local variable within the same Script Task)
myString = "myValue";
// a date object can be created from context variables
var myDate = new Date($.context.myDate);
// persisting the date back to the context will store it in ISO 8601 format
$.context.myNewDate = myDate;
```

Removing Variables

```
// the following will remove the variable from the context
delete $.context.myString;
```

Manipulating the Context Directly

```
// The workflow context can be cleared completely. The $.context API will
continue to exist, but all variables will have been removed.
$.context = null.
// The workflow context can be completely overwritten, by setting it to an
object, whose properties are becoming the new context variables.
$.context = {newField: "new value"};
```

Complex structures can be, for example, objects and arrays and you can create and use to manipulate such structured data. For more information, see the *Related Links*.

Related Information

Modifying the Workflow Context with Objects [page 54]
Modifying the Workflow Context with Arrays [page 55]

3.1.3.1.3.1.1 Modifying the Workflow Context with Objects

You can insert scripts to modify the workflow context, for example, to transform data from one representation to another, and also to read and set values.

For working with objects in JavaScript, the following sample scripts are available:

Constructing Objects

```
// Create a new object with a simple property and persist it
$.context.myObject = {newField: "new value"};
// You can also assign a local object
var obj = {newField: "new value"};
$.context.myObject2 = obj;
```

Object Property Access

```
//the following access to objects and their properties are equal
var myObject = $.context.myObject;
var prop = myObject.myProperty;
var prop2 = $.context.myObject.myProperty;
// Objects are accessed by "reference", myNumber will be stored directly in the
workflow context
// the local variable 'myNumber' will have the value 42
var obj = {newField: "new value"};
$.context.myObject2 = obj;
obj.myNumber = 42;
var myNumber = $.context.myObject2.myNumber
```

Object Conversions

```
var prop = $.context.myObject.myProperty;
if (typeof prop === 'number') {
// ... use JavaScript data type conversions
}
else if (typeof prop === 'object') {
var propAsInt = parseInt(prop.stringProperty); // for example, "42"
}
```

3.1.3.1.3.1.2 Modifying the Workflow Context with Arrays

You can insert scripts to modify the workflow context, for example, to transform data from one representation to another, and also to read and set values.

For working with arrays, the following sample scripts are available:

Constructing Array

```
// Create a new array with three entries
var array= ["one", "two", "three"];
array.push("four");
$.context.myArray = array; // stores array [one, two, three, four] in the context
array.push("five"); // adds five to the context
```

Manipulating Array

```
// Insert entries into array at specific positions
var array = $.context.myArray;
if (array.length == 0) {
```

```
array.push("first"); // adds a new element at the end of the array
array.splice(1,1); // removes entry at position 1, the one that was previously
the first
array.unshift("new first"); // adds a new element at the beginning of the array
// array.splice(-1, 1); // out of bounds deletions
// array.splice(42, 1); // resp. at the last position
}
if (array.length == 1) {
var el = array.shift(); // returns the first element of the array and deletes it
from the array
array.unshift("new first"); // adds a new element at the beginning of the array
}
var idx = array.indexOf("new first"); // returns the index of the first
occurrence of the passed value
// all JavaScript ECMA 5.1 array functions are supported (http://ecma-
international.org/ecma-262/5.1/)
```

Array Index Access

```
var arr = $.context.myArray;
var entry = arr[0]; // first entry in array
```

3.1.3.1.3.2 Accessing Contextual Information During Execution of Script Tasks

You can insert scripts to allow access to identifiers of the current task or the exact execution. Unique identifiers are, for example, necessary to propagate calls to external services.

Getting Information About the Environment

Technical ID	Technical Type	Sample Value
workflowInstanceId	String	336963b0-3726-49fa-bf0c-87a8f7aa-baf8
workflowDefinitionId	String	onboardingprocess
startedBy	String	John
businessKey	String	ONBOARDING-247

```
var workflowInstanceId = $.info.workflowInstanceId ; // for example,
"336963b0-3726-49fa-bf0c-87a8f7aabaf8"
var workflowDefinitionId= $.info.workflowDefinitionId; // for example,
"onboardingprocess"
var startedByUserId = $.info.startedBy; // for example, "John"
var businesskey = $.info.businessKey; // for example, "ONBOARDING-247"
```

Getting Information About User Task Instances

To allow access to properties of user task instances, you can insert scripts. Use the `$.usertasks` object as an entry point followed by the user task definition ID from the workflow model: `$.usertasks.<User Task Definition ID>`. For example, if the ID of a user task is `usertask1`, then use `$.usertasks.usertask1.last.priority` to point to the priority of the instance of the corresponding task definition, which was created last.

The following properties are available for the objects that refer to user task instances:

Property Name	Type in Script Task	Comments
id	String	-
createdAt	Date*	-
createdBy	String	-
priority	String	-
dueDate	Date*	-
status	String	-
subject	String	-
description	String	-
recipientUsers	Array of strings	-
recipientGroups	Array of strings	-
processor	String	Only for task status RESERVED or COMPLETED
claimedAt	Date*	Only for task status RESERVED or COMPLETED
completedAt	Date*	Only for task status COMPLETED

* Please note that dates are represented as strings in expressions. For more information, see Expressions [page 78].

In the following code snippet, the processor of the last created instance of the usertask1 is written into the context variable `taskProcessor`.

```
$.context.taskProcessor = $.usertasks.usertask1.last.processor;
```

Script tasks cannot modify the `$.usertasks` API. All its properties are provided by SAP Workflow service and are read-only.

Saving Contextual Information in the Workflow Context

You can save an object that refers to the last instance of a user task in the workflow context.

```
$.context.lastUserTask1 = $.usertasks.usertasks1.last;
```

So, at the time a script is executed, a snapshot of the last user task instance is created and persisted in the context.

Please note that, as of now, this is the only complex nested property of the `$` object that can be stored in the workflow context.

If you try to save one of the following objects into context, an error occurs when the workflow instance is executed.

```
$.context.variable = $.context;
$.context.variable = $.info;
$.context.variable = $.usertasks;
$.context.variable = $.usertasks.usertasks1;
```

There is no limitation on saving primitive values in the workflow context and the following code is absolutely valid:

```
$.context.variable = $.info.workflowInstanceId;
```

3.1.3.1.3.3 Get and Set Instance-Specific Roles

In script tasks, you can access instance-specific roles of the current workflow instance.

Use the `$.roles` object as an entry point followed by the type of the role you want to read from or write to. For example, in a script task for a given workflow instance, you can access the current list of admin users through `$.roles.adminUsers`.

The following variants are available for the `$.roles` object that refers to the instance's roles:

Type	Script Task Object/Property (Read/Write Access)	JUEL Expression (Read-only)
Array of Strings of viewer users	`$.roles.viewerUsers`	`${roles.viewerUsers}`
Array of Strings of viewer groups	`$.roles.viewerGroups`	`${roles.viewerGroups}`
Array of Strings of context viewer users	`$.roles.contextViewerUsers`	`${roles.contextViewerUsers}`
Array of Strings of context viewer groups	`$.roles.contextViewerGroups`	`${roles.contextViewerGroups}`
Array of Strings of admin users	`$.roles.adminUsers`	`${roles.adminUsers}`

Type	Script Task Object/Property (Read/Write Access)	JUEL Expression (Read-only)
Array of Strings of admin groups	`$.roles.adminGroups`	`${roles.adminGroups}`
Array of Strings of context admin users	`$.roles.contextAdminUsers`	`${roles.contextAdminUsers}`
Array of Strings of context admin groups	`$.roles.contextAdminGroups`	`${roles.contextAdminGroups}`

> **Note**
>
> If no user or group is or should be set for the given role, an empty array is used.

Assign Roles to User

In this example, you assign the admin role to the user Julie.

Sample Code

```
var admins = $.roles.adminUsers;
admins.push('Julie');
```

In this example, you assign the viewer role to the users John, Michael, and Richard.

Sample Code

```
var viewers = ['Michael', 'Richard'];
$.roles.viewerUsers = viewers;
// ...
viewers.push('John');
```

Retrieve Users Assigned to a Role

In this example, you can read which users are assigned to the viewer role.

Sample Code

```
var instanceViewers = $.roles.viewerUsers; // for example, ["John",
"Michael", "Richard"]
// single access
if (instanceViewers.length > 0) {
    var firstViewer = instanceViewers[0]; // for example, "John"
    // do something with firstViewer
}
// iteration
```

```
instanceViewers.forEach(function (viewer) {/* do something with 'viewer', for
example, John, Michael, and then Richard */});
```

Unassign Users From Roles

In this example, you unassign all users from the viewer role.

Sample Code

```
$.roles.viewerUsers = []; // clears only viewer users, but not any other roles
```

3.1.3.1.4 Configure Mail Tasks

A mail task is a flow object that can be configured to send e-mails to one or more recipients.

Prerequisites

Configure a mail destination. See Configure the Workflow Service Mail Destination [page 177].

Procedure

1. Choose  (Tasks), then *Mail Task* from the palette and drop it on to the canvas.
2. In the *Mail Task Properties* area, choose the *General* tab.
3. Provide a *Name* and *Documentation*.

   > **Note**
   >
   > A unique, read-only ID is generated for every workflow artifact.

4. From *Mail Task Properties* area, choose the *Details* tab.
5. In the *To* field, provide a list of comma-separated mail addresses to which to send the mail.

   > **Note**
   >
   > For more information about character limits for SAP Workflow service, see Conventions, Restrictions, and Limits [page 8].

6. (Optional) Add mail addresses to the *Cc* for the mail and *Bcc* fields.

> **Note**
>
> If there is a syntax error in any of the *To*, *Cc*, or *Bcc* fields, the mail task execution is aborted and the workflow instance changes to an error status.

7. Choose *Ignore Invalid Recipients* to ignore any invalid e-mail addresses.

> **Note**
>
> - If you select this option, e-mail addresses that are syntactically incorrect or that are caused by unresolvable expressions won't cause the task to fail, provided at least one recipient can be determined. The ignored recipients list appears in the *Monitoring Workflows* app
> - If the option is disabled, mail task fails when at least one invalid recipient is determined.

8. Provide a *Subject*

> **Note**
>
> *Subject*, *Cc*, *Bcc*, and *To* fields can contain JUEL expressions. For more information, see Expressions [page 78].
>
> Except for the *Subject* field, you can use either a context reference that resolves to a string, with different mail addresses separated by commas, or a context reference that resolves to an array of mail addresses.

9. From the *Configure Mail Body* list, choose one of the following:
   - *Plain Text*: Provide the message in the form of text.
   - *HTML*: Create a new or choose an existing HTML file for the mail content.
     To create a new HTML file, perform the following steps:
     1. In the *HTML Body* section, choose the *Create file* link.
     2. In the *Create New File* window, provide a file name.
     3. Choose *Create*.
     4. In the HTML file, provide the mail content.

        > **Note**
        >
        > - If you have set the *Mail Body* to HTML, the text representation of the emails that are sent is derived from the HTML content and specified as an alternative representation of the HTML content in the e-mail. E-mail clients typically display the text representation in text-only mode. However, it is at your discretion to use text-only mode.
        > - In many cases, the derived text is suitable to be shown to end users. However, in cases of complex HTML structures, the text representation might not be optimal. If the text representation is important to you, simplify the HTML code to use mostly simple, semantic mark-up or specify the mail body directly as text.
        > - You can use expressions in the mail body and the subject. However, you cannot add HTML tags in the HTML mail body using expressions, because special characters in the expression results are escaped for security reasons.
        > - An example HTML mail body is shown below.
        >
        > **Sample Code**
        >
        > ```
        > <!doctype html>
        > <html>
        > ```

```
    <head>
        <title>Workflow Service Email Notification</title>
        <style>
            h3 {
                font-family: serif;
            }
            p, dl, dd, dt {
                font-family: sans-serif;
            }
            dt {
                text-indent: 5em;
            }
        </style>
    </head>
    <body>
        <h3>Employee onboarding completed</h3>
        <p>Dear ${context.initiatorName},</p>
        <p>The employee onboarding that you triggered has been
successfully completed.</p>
        <p>Sincerely yours,</p>
        <p>SAP Workflow service</p>
    </body>
</html>
```

5. Save and close the HTML file.

> **i Note**
>
> - You can find the HTML file in the following location: `<workflow-project>/webcontent/<workflow-name>/<html-file-name>.html`.
> - The list of e-mail addresses and JUEL expressions can contain a maximum of 5000 characters and 100 e-mail addresses.
> - *Subject* can be a maximum of 1000 characters long. We recommend that you use far fewer characters because mail clients can show only a limited subject length.
> - *Mail Body* can contain a maximum of 10000 characters.

10. Connect the mail task to the required flow elements.
11. Choose *Save*.

Related Information

Conventions, Restrictions, and Limits [page 8]

3.1.3.1.5 Configure a Subflow

You want to reference a workflow that is executed inside another (main) workflow.

Prerequisites

You have created a workflow in which you want to add the subflow.

Context

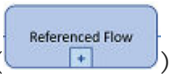
Inside one workflow you can execute another workflow called a referenced subflow. You can configure the activity with which you call the subflow. The subflow is an external workflow definition that can be reused and is deployed independently from the main workflow.

The main workflow stores data in the workflow context. You can pass a subset of this data to the subflow by configuring data mapping. The subflow can modify the data. Therefore, it can return the data back into the main workflow using data mapping. The data is copied into the subflow when it is started and copied back into the main worflow when the subflow ends.

Main workflow calls the subflow and waits until the subflow ends before it continues with the next item. The workflow definition of the subflow is resolved at runtime. This means, that the subflow can be deployed independently of the calling main workflow.

The subflow can be a workflow from the same project as the main workflow, or it can be any workflow that is deployed on the server.

Procedure

1. Choose  (Tasks), then *Referenced Subflow* from the palette and drop it on to the canvas.
2. Select the referenced subflow icon (

) that you dropped on the canvas.
3. In the *Referenced Subflow Properties* area, choose the *DETAILS* tab.
4. Enter the workflow definition ID of the workflow you want to use as a subflow.

   Either type in the ID as text or use a JUEL expression like `${context.propname}`.
5. Optional. To start the subflow you can use a different user than the one used in the main workflow:
   a. Select *Principal Propagation*.
   b. On the *Select Flow Element* popup, choose the element that contains the user you want to propagate to the subflow.
6. Choose the *MAPPING* tab, and add the object information in the tables using variables, for example, `${context.product}`.

The mapping is used to transfer data between the main workflow and the subflow. There are two transfer directions:

- Input Mapping: Define data passed on to the subflow
  On the start of the subflow, data flows from the main workflow to the subflow. The *Source Context Path* contains the process context of the main workflow. It is mapped to the *Target Context Path* that contains the process context of the subflow.
- Output Mapping: Define data passed back from the subflow to the main workflow
  When the execution of the subflow ends, the data is transferred back to the main workflow. Now, the source and target are switched around: "Source" is the subflow and "Target" is the main workflow.

7. Choose *Save*.

Related Information

Referenced Subflow – Modularize your workflows

3.1.3.2 Events

An event affects the flow of the process.

SAP Workflow service editor supports the following events:

Start event: It indicates where a workflow starts and what triggers a workflow. Start events have no incoming sequence flow. Each workflow has one start event.

Intermediate Message Event: Intermediate message events are process steps where the respective workflow instance waits for a message before the flow commences in the respective control flow branch.

Intermediate Timer Event: It allows a workflow to pause and resume after a specified interval of time.

End event: An end event means that this event has no specific result. End events have no outgoing sequence flow. Consider a workflow that has several branches, the workflow terminates only after all the branches gets executed.

Terminate end event: The terminate event ends the workflow in a regular way. But, consider a workflow consists of multiple branches and you choose one branch as a terminate end event. The workflow terminates when the branch marked as terminate end is executed without waiting for other branches to get executed.

3.1.3.2.1 Configure Start Events

You can configure a sample context while modeling a start event.

Prerequisites

You are in the process of modeling a start event. For more information, see Define Workflows [page 26].

Context

Alternatively, you can use the API to retrieve the sample start context. For more information, see Using Workflow APIs [page 151].

You can propagate the user who starts a workflow instance to a service called later by a service task in the same workflow instance. For more information, see Configure Service Tasks [page 45].

Procedure

1. Choose the *Details* tab of the start event.
2. Select the *Configure sample context* checkbox.
3. To browse for a JSON file in the current project, choose *Select*.
4. To create a JSON file, perform the following substeps:
   a. Choose the *Create file* link.
   b. In the *Create New File* window, provide a file name with `.json` extension.
   c. Choose *Create*.
   d. In the JSON file, provide the context.

   > **i Note**
   > - You can view or edit the JSON file by selecting the *File* link.
   > - For more information about *Code Editor*, see Develop Your Application.
5. Save the workflow.

Related Information

Define Workflows [page 26]

3.1.3.2.2 Configure Intermediate Message Events

Intermediate message events occur when a workflow instance waits for a message before the flow commences in the respective control flow branch.

Prerequisites

Configure a business key for your workflow. For more information about business keys, see Define Workflows [page 26].

Context

Clients can send messages using the REST endpoint. For more information about how to send messages, refer to Workflow Service API documentation at Using Workflow APIs [page 151].

The messages received through this endpoint are synchronously correlated to workflow instances based on the business key. The message can be delivered to one or more instances of the same workflow definition, which has a matching business key and an active execution branch waiting at the intermediate message event.

Procedure

1. Select ○ *Events* > *Intermediate Message*, and drop it onto the canvas from the palette.
2. In the *Intermediate Message Event Properties* area, choose the *General* tab.
3. Fill in the *Name* and *Documentation* fields for the intermediate message event.
4. In the *Intermediate Message Event Properties* area, choose the *Details* tab.
5. In the *Message Name* field, provide a name of the message.

   > **i Note**
   >
   > For more information about character limits for SAP Workflow service, see Conventions, Restrictions, and Limits [page 8].

6. (Optional) Provide a *Response Variable* link to a workflow context node, which holds the context data passed by the incoming message.

   > **i Note**
   >
   > - If you use a response variable, it must adhere to the syntax defined by the Java Unified Expression Language (JUEL). You can only use expressions that reference the workflow context. For more information, see Expressions [page 78].
   > - If you don't provide a response variable, the message is consumed by matching workflow instances. However, the context data passed by the message is not considered.

7. Choose *Save*.

> **Example**
>
> Equipment must be procured for a new hire. In this case, the employeeID of the new hire can be configured as business key. The workflow calls an external service to trigger the asynchronous procurement process. The workflow instance must wait until the procurement process is completed.
>
> You can model an intermediate message event, which blocks the execution of the workflow in this branch until a message is received. When the procurement process completes, the external system can send a message that includes details about the equipment ordered. This message is then delivered to one of the waiting workflow instances, and the execution moves to the next flow step.

Related Information

Conventions, Restrictions, and Limits [page 8]

3.1.3.2.3 Configure Intermediate Timer Events

Configure an intermediate timer event to allow a workflow to pause and resume after a specified interval of time.

Context

In a few business scenarios, a workflow may need to wait for a certain interval of time before proceeding with the flow; for example, a workflow that updates multiple systems of record. You can add an intermediate timer event that delays the workflow for a few minutes, to ensure that all records have been updated before the workflow continues.

Procedure

1. Select ○ *Events* > *Intermediate Timer* and drop it onto the canvas from the palette.
2. Fill in the *Name* and *Documentation* fields for the intermediate timer event.
3. In the *Intermediate Timer Event Properties* area, choose the *Details* tab.
4. Provide the waiting time interval in the *Duration* field.
   - To use expressions, choose *Expression* from the *Duration Based On* dropdown.

   > **Note**
   >
   > Provide an expression in the *Duration* field using ISO 8601 format. For example, `PT${context.minutes}M`. The JUEL expression `${context.minutes}` is evaluated at runtime. You

> can provide multiple duration attributes by using multiple JUEL expressions. For more information about the duration formats that are supported in ISO 8601, see Conventions, Restrictions, and Limits [page 8].

   - To use a static value, choose *Static Value* from the *Duration Based On* dropdown. Now, provide the *Duration* as a numeric value, and choose a *Unit of Time*.

5. Choose *Save*.

3.1.3.3 Gateways

A gateway controls the flow of execution and is represented visually as a diamond shape with an icon inside. The icon shows the type of gateway.

The SAP Workflow service editor supports the following gateway types:

- **Exclusive gateway**: Use an exclusive gateway to model a decision in the process. When the execution arrives at this gateway, all outgoing sequence flows are evaluated in the order in which they're defined. The sequence flow with a condition that evaluates to true is selected for continuing the process.
  If multiple sequence flows have a condition that evaluates to true, the first one defined is selected for continuing the process. If none of the conditions defined for the sequence flow evaluate to true, then the one marked as default flow is selected and the execution proceeds along that path.

  > **i Note**
  >
  > If you use an exclusive gateway to split the flow into multiple sequence flows, then also use an exclusive gateway to merge the flows again. Otherwise, the workflow instances get stuck at the parallel join.

  For more information, see Configure an Exclusive Gateway [page 69].
- **Parallel gateway**: Use a parallel gateway to split into multiple paths of execution or merge multiple incoming paths of execution. The functionality of the parallel gateway is based on the following incoming and outgoing sequence flow:
  - Split: All outgoing sequence flows are executed in parallel; there's one concurrent execution for each sequence flow.
  - Join: All concurrent executions arriving at the parallel gateway wait in the gateway until an execution has arrived for each incoming sequence flow. Then the process continues past the joining gateway.

  > **i Note**
  >
  > A parallel gateway works on a logical level. It doesn't speed up the technical execution.

  For example, consider a scenario where an employee approaches the travel desk to book a flight and a hotel accommodation for a business trip. With a parallel gateway, both the flight arrangement and the hotel accommodation can happen in parallel. Once the booking is successful, an email notification can be sent to the employee.

  > **i Note**
  >
  > An exclusive gateway can be used to merge multiple parallel paths. In this case, however, the workflow instance does not wait at the join for all the executions to arrive into the incoming sequence flows. Instead, the part of the workflow after the join is executed for each of them right away as soon as they

arrive. This pattern can be used in cases when exactly the same steps must be executed in multiple parallel branches.

For more information, see Configure a Parallel Gateway [page 70].

3.1.3.3.1 Configure an Exclusive Gateway

You use this procedure to configure an exclusive gateway in workflow editor.

Procedure

1. From the palette, choose ◇ *Gateways* *Exclusive Gateway* drop it on to the canvas.
2. In the *Exclusive Gateway Properties* area, provide a *Name* and *Documentation* for the gateway.

   > **i Note**
   >
   > A unique *ID* gets generated for every workflow artifact. This ID is in read-only mode.

3. On the canvas, create a sequence flow from the *Exclusive Gateway* icon to other flow objects.

   > **i Note**
   >
   > If there are more than one outgoing sequence flows from an exclusive gateway, then it is considered as a split in the flow. Only in this case, you can view and configure the *Sequence Flow Properties* . The next step of configuring a condition is only possible in case of a split scenario.

4. Configure a condition for a sequence flow.

   For more information, see Configure a Sequence Flow [page 70].
5. Choose *Save*.

Related Information

Accelerated Modeling with Speed Buttons [page 76]

3.1.3.3.1.1 Configure a Sequence Flow

Procedure

1. On the canvas, choose the sequence flow you want to configure.
2. In the *Sequence Flow Properties* section provide a *Name* and *Documentation* for the flow object .

   > **i Note**
   >
   > A unique *ID* gets generated for every workflow artifact. This ID is in read-only mode.

3. From an exclusive gateway, provide a *Condition* to the outgoing sequence flow or mark the sequence flow as *Default*.

   > **i Note**
   >
   > - You can mark only one outgoing sequence flow as the default.
   > - If you want a certain path to execute, for example, only if an employee does not belong to Germany. You need to configure the sequence flow condition as *${context.employee.region!= "Germany"}*. For more information, see Expressions [page 78].

4. Choose *Save*.

3.1.3.3.2 Configure a Parallel Gateway

You use this procedure to configure a parallel gateway in workflow editor.

Procedure

1. From the palette, choose ◇ *Gateways* *Parallel Gateway*, and drop the icon on to canvas.
2. In the *Parallel Gateway Properties* area, provide a name and documentation for the gateway.

   > **i Note**
   >
   > A unique *ID* gets generated for every workflow artifact. This ID is in read-only mode.

3. If you are creating a split, then create multiple outgoing sequence flows from the parallel gateway.
4. If you are creating a join, then create multiple incoming sequence flows to the parallel gateway.
5. Choose *Save*.

Related Information

Accelerated Modeling with Speed Buttons [page 76]

3.1.4 Configure Custom Workflow Attributes

You can assign custom workflow attributes to your workflow.

Context

With custom workflow attributes, you can define business-related properties and assign them to workflows such as project ID or project name.

At runtime, you can use the respective workflow service API to search for custom workflow attributes or to find the respective workflow instances. For more information about the characteristics of the various APIs, see Using Workflow APIs [page 151].

Procedure

1. Choose the *Attributes* tab.
2. To add a row, choose *Add*.

   You can reorder the added custom workflow attributes by using *Move Up* or *Move Down*.
3. Provide the following details in the table:

Name	Description
ID	A unique identifier of the attribute within a workflow. **i Note** You can only use alphanumeric characters as ID and it must only start with an alphabet.
Label	A human readable name of the attribute, which appropriate user interfaces can use to label the attribute.
Type	Data type of the attribute. **i Note** Currently, only the data type string is supported.

Name	Description
Value	A constant, a JUEL expression, or a mixture of constants and JUEL expressions. It gets resolved upon workflow creation and on every change to the context, for example, through service or script tasks. For JUEL expressions, only the `${context.xyz}` subset is supported. `${info}`, `${roles}`, and `${usertasks}` aren't supported. **Example** `USR_${context.user.userId}` A complete array isn't resolved in JUEL expressions. For `${context.aArray}`, for example, values of that kind of expression always resolve to null. However, for a single array element it resolves, for example, `${context.aArray[0]}`. A complex object is also not resolved in JUEL expressions. For `${context.complexObject}`, for example, values of that kind of expression always resolve to null. However, for a single object element it resolves. For example, `${context.complexObject.primitiveProperty}` resolves to the value of the primitive property. `${context}` doesn't contain any node. It's also not supported and its value always resolves to null.

Note

A workflow can contain up to 15 attributes at a time. For more information, see Conventions, Restrictions, and Limits [page 8].

4. Save the changes.

3.1.5 Translate Workflows

You need to enable translation for a workflow before it can be translated.

Prerequisites

- You are using the SAP Business Application Studio to develop applications. See Developing Applications with Workflow Service [page 21].
- You have created an MTA with a workflow module. See Generate an MTA Project Containing a Workflow Module with SAP Business Application Studio [page 23].

Context

Some texts within a workflow support translation, for example, the `User Task Subject`. Translation texts are maintained in the resource bundles of a workflow. For the developer language, the respective resource bundle (`i18n.properties` file) is automatically kept in sync with its workflow. This means, texts defined for a workflow in the workflow editor are stored in the `i18n.properties` file.

For each language, the texts need to be translated and stored inside a new resource bundle. For example, the German translations are stored in a `i18n_de.properties` file right next to the `i18n.properties` file.

Translations are considered in My Inbox for user tasks, in the Monitor Workflows apps, and in the REST APIs by providing an `Accept-Language`. See the Workflow API for Cloud Foundry.

> **i Note**
>
> To add new languages or to change existing texts, a new MTA build and deployment is needed. New translations are stored once the MTA is deployed.

Translation-Enabled Artifacts

Workflow Elements	Translation-Enabled Artifacts	Supported Properties
Workflow		Name
		Subject
Event	Start Event	Name
	Intermediate Message	
	Intermediate Timer	
	End Event	
	Terminate End Event	
Tasks	User Task	Name
	Service Task	Subject (only user task)
	Mail Task	Description (only user task)
Gateways	Parallel Gateway	Name

Procedure

1. In SAP Business Application Studio, access your workflow.
2. On the *Translation* tab, choose *Enable Translation*.

   This automatically creates a resource bundle for the workflow.

3. Model your workflow.

   For translation-relevant properties in the workflow, the values are automatically kept in sync with your `i18n.properties` file (developer language).
4. Build and deploy your workflow. Build and Deploy the Workflow Module [page 75]

Notes

- If the workflow is already enabled for translation, the link to the developer resource bundle `i18n.properties` is displayed instead of the *Enable Translation* button.
- If the workflow already contains modeled artifacts when you enable it for translation, the `i18n.properties` file takes the existing artifacts into account.
- The resource bundle containing the developer texts is located at `<mta root>/<workflow module>/i18n/<my workflow name>/i18n.properties`.
- The My Inbox app, Monitor Workflows app, and the workflow service APIs return translated texts for translation-relevant properties if a translation exists for the logon language of the user, for example, `en_US`. If no translation for the logon language exists, a fallback language is determined. If no fallback exists, the texts as defined in the workflow definition are returned.
- Any manual change in the `i18n.properties` file, is overwritten when you save the respective workflow.

Restrictions

- Translation of the mail task subject and body is currently not supported.
- Translations for values within the workflow context are not supported out-of-the-box.
- Only translation files of the SAP's top 8 languages are considered and only translations for the supported properties.

3.1.6 Transport Workflows between Accounts

SAP Cloud Transport Management in the SAP BTP, Cloud Foundry environment helps in transporting the content from one sub account to another one.

Workflows can be modelled as part of a multitarget application (MTA) project in the SAP BTP, Cloud Foundry environment. You can build the MTA project that generates the `.mtar` file containing application modules. Using the SAP Cloud Transport Management, you can transport all the modules that are part of the MTA project to the target sub accounts as part of one transport action. For more information, see What Is Cloud Transport Management.

> **i Note**
>
> The transport of a workflow module checks the uniqueness of a workflow definition ID across all service instances and subscriptions. See Conventions, Restrictions, and Limits [page 8].

3.1.7 Build and Deploy the Workflow Module

Prerequisites

You're assigned the *Space Developer* role. For more information, see Initial Setup [page 17].

> → Recommendation
>
> - Create a workflow instance and then reuse in the `mta.yaml` file.
> - Enable parallel deployment of application modules in the multitarget application (MTA) deployment descriptor (mta.yaml). For more information, see *Parallel Deployment* section in Order of Deployment and Modules.

Context

To deploy your workflow module along with workflow resources, you must build and deploy the containing project.

Procedure

1. Save any workflow resources you've modified.
2. Modify the multitarget application (MTA) module types by providing either of the following resources types and parameters in the MTA deployment descriptor (`mta.yaml`).
   - To use an existing instance of workflow service, define it by using `org.cloudfoundry.existing-service` resource type along with the existing resource name.

     > → Recommendation
     >
     > Use this approach to modify the descriptor (`mta.yaml`) file.

     You can find the existing instance name of workflow service by navigating to *Services* > *Service Instances* in the cockpit.

     > ‘≒, Sample Code

     ```
     resources:
       - name: <existing_workflow_service_instance_name>
         type: org.cloudfoundry.existing-service
     ```

   - If there's no workflow service instance created, create a new workflow service instance. Define it using the `org.cloudfoundry.managed-service` resource type along with the resource name provided by the user.

Sample Code

```
resources:
  - name: <workflow_service_instance_name>
    type: org.cloudfoundry.managed-service
    parameters:
      service-plan: standard
      service: workflow
```

For more information about resources types and parameters, see MTA Module Types, Resource Types, and Parameters for Applications in the Cloud Foundry Environment.

3. Add dependency to the workflow service instance in the `requires` section of the workflow module.
4. Build and deploy your project.

   For more information, see the following:

   With SAP Web IDE: Packaging and Deploying Applications to Production Systems

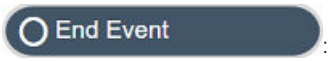
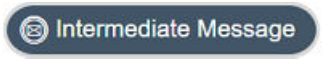
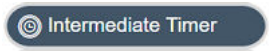
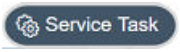
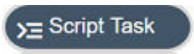
   With SAP Business Application Studio: Build Your Application and Deploy Your Application

> **Note**
>
> The deployment of a workflow module checks the uniqueness of a workflow definition ID across all service instances and subscriptions. See Conventions, Restrictions, and Limits [page 8].

3.1.8 Accelerated Modeling with Speed Buttons

In addition to the palette, you can use the speed buttons for quick and easy modeling. The speed buttons are displayed around the flow objects. In the following figure you see a start event with the speed buttons around it. The number and type of speed buttons that are displayed vary depending on the model element.

The following table contains all the different types of speed buttons and explains their function:

Speed Button	Description
	This speed button allows you to model the following events: •  : Creates an end event. •  : Creates an intermediate message event. •  :Creates an intermediate timer event.
	This speed button allows you to model the following tasks: •  : Creates a user task. •  : Creates a service task. •  : Creates a script task. •  : Creates a mail task.
	This speed button allows you to model the following gateways: •  : Creates an exclusive gateway. •  : Creates a parallel gateway.
	Creates a sequence flow from one flow element to another. **i Note** When you select the sequence flow speed button, you must drag it on the element that you want to connect to. If the area is highlighted in green, then the element can be connected using a sequence flow. If the area is not highligted, then the element cannot be connected using the sequence flow.
	Allows you to create a boundary timer event on a user task.
	Allows you to delete a flow element.

3.1.9 Expressions

There are several places in the editor where you can enter expressions to extract data from the workflow context.

Expressions are mainly used for the following purposes:

- To combine static texts and variables. These are, for example, shown as texts to the user to provide contextual information (text expressions).
- To determine major task properties dynamically (property navigation)
- To determine the next steps when the control flow arrives at gateways (conditions)

The expressions you use must adhere to the syntax defined by the Java Unified Expression Language (JUEL). You can access data stored in the workflow context, for example, `${context.variablename}`) as well as data that refers to the current task or workflow, for example, `${info.workflowInstanceId}`). The syntaxes to access this data within a JUEL expression and using the script task API are aligned. The following statements address the same attribute:

Functionality	Example of JUEL Expression	Example of Script Task API	More Information
Workflow instance context	`${context.employee.firstname}`	`$.context.employee.firstname`	Creating and Reading Workflow Context Structures [page 52]
Workflow instance information	`${info.workflowInstanceId}`	`$.info.workflowInstanceId`	Getting Information About the Environment in Accessing Contextual Information During Execution of Script Tasks [page 56]
Workflow instance roles	`${roles.adminGroups}`	`$.roles.adminGroups`	Get and Set Instance-Specific Roles [page 58]
User task properties	`${usertasks.<User Task Definition ID>.last.processor}`	`$.usertasks.<User Task Definition ID>.last.processor`	*Getting Information About User Task Instances* in Accessing Contextual Information During Execution of Script Tasks [page 56]

For a detailed overview of the JUEL expressions and the corresponding script task API, see Cheat Sheet for Workflow Expressions [page 82].

Property Navigation and Text Expression

Property navigation and text expressions typically occur in user tasks. See Configure User Tasks [page 29].

Example

Sample Code

```
{
    "context": {
        "employee": {
            "name"  : "Peter",
            "peers" : [
                {
                    "name": "Mary"
                }
            ],
            "region": "Germany",
            "userId": "9899"
        }
    }
}
```

Examples that are supported by expression syntax include the following:

- Property navigation (including text expressions without static texts)
  - Accessing the `name` property of the `employee` context variable (dot notation): `${context.employee.name}`
  - Accessing the `name` property of the first entry in the `peers` array property of the `employee` context variable: `${context.employee.peers[0].name}`
- Text expressions
  Combining static with dynamic content: `Dear ${context.employee.name}`
- Conditions
  Applying Boolean operators to form a condition:
  `${context.employee.region!= "Germany" && context.employee.isManager == true}`

Conditions and Variable Specifications

Besides the already described types of expression, there are several other types:

- Condition expressions have to evaluate to a Boolean value, that is true or false.
- You must specify conditions like the ones used in the example in Configure a Sequence Flow [page 70] as follows:
  `${context.employee.region!= "Germany" && context.employee.isManager == true}`
- You must specify the variable to retrieve the request body when calling external services (see Configure Service Tasks [page 45]) as follows:
  `${context.employee.oldEmpData}`
- Expressions that create new structures within context variables support only the dot notation of the JUEL language. You must, for example, specify the expression to store the response from an external service (see Configure Service Tasks [page 45]) as follows:
  `${context.employee.empData}`

Notices

- When there are multiple expressions in a single field: if one of the expressions is incorrect or refers to a field that does not exist, then none of the expressions in that field are replaced. For example, in the text expression "`Approval for ${context.employee.firstname} ${context.employee.lastname}`", if the employee's last name field does not exist, none of the expression is replaced.
- Once expressions in texts are resolved, that is, they are replaced with the actual text at runtime, the texts are not changed if the process context changes at later point in time.

Overview of Properties Supporting JUEL Expressions

> **i Note**
>
> All task-related properties of `${info}` are only available on JUEL-enabled properties of service and user tasks.

Elements and Properties Using JUEL Expressions

Workflow Model Element Type	JUEL-Enabled Property	Required Data Type
Sequence flow originating from an exclusive gateway	Condition	Boolean
Service task	Path	String
	Path to XSRF token	String
	Destination	String
User task	Subject	String
	Recipient users	Comma-separated string or array of strings
	Recipient groups	Comma-separated string or array of strings
	HTML5 app	String
	Component URL	String
	SAPUI5 component	String
Mail task	To	Comma-separated string or array of strings

Workflow Model Element Type	JUEL-Enabled Property	Required Data Type
	Cc	Comma-separated string or array of strings
	Bcc	Comma-separated string or array of strings
	Subject	String
	HTML / plain text body	String
Workflow model	Subject	String
	Business key	String

Handling of the Date-Related Objects

Dates are handled differently in script tasks and in expressions. In script tasks, the JavaScript date is used to represent date-related properties of workflow service APIs, for example, the `createdAt`, `claimedAt`, and `completedAt` properties of user tasks. However, in expressions, the corresponding properties are represented as strings.

In addition, all values saved in the context as dates in script tasks are converted to the corresponding strings at the end of the script task execution. They are available as strings in subsequent JUEL expressions and script tasks.

Related Information

JUEL Tutorial - Expression Language

3.1.9.1 Cheat Sheet for Workflow Expressions

This overview lists the APIs available for the use in JUEL expressions and script tasks.

Creating and Reading Workflow Context Structures

Type and Description	Script Task Object/Property Examples (Read/Write Access)	JUEL Expression Examples (Read-only)	More Information
Reading variables from context	`var myAlias = $.context.myString;`	`${context.myString}`	Creating and Reading Workflow Context Structures [page 52] Commonly used operations of the EcmaScript 5.1 / JUEL are supported
Setting variables in context	`$.context.myString = "Hello";` `$.context.myNumber = 314;`	Not possible	n.a.
Removing variables from context	`delete $.context.myString;`	Not possible	n.a.
Reading array lengths	`var count = $.context.myArray.length;`	`${context.myArray.length}`	n.a.

Writing arrays	`$.context.myArray = [];`
	`$.context.myArray = ["1","2","3"];`
	`$.context.myUserArray = $.usertasks.usertask1.recipientUsers;`
	`// see below`
		Not possible	n.a.

Writing dates into context	`$.context.currentDate = new Date(1000000000000);` `// stores "2001-09-09T01:46:40.000Z"`	Not possible	n.a.

Get and Set Instance-Specific Roles

Type and Description	Script Task Object/Property (Read/Write Access)	JUEL Expression Examples (Read-only)
Array of Strings of viewer users	`$.roles.viewerUsers`	`${roles.viewerUsers}`
Array of Strings of viewer groups	`$.roles.viewerGroups`	`${roles.viewerGroups}`
Array of Strings of context viewer users	`$.roles.contextViewerUsers`	`${roles.contextViewerUsers}`
Array of Strings of context viewer groups	`$.roles.contextViewerGroups`	`${roles.contextViewerGroups}`
Array of Strings of admin users	`$.roles.adminUsers`	`${roles.adminUsers}`
Array of Strings of admin groups	`$.roles.adminGroups`	`${roles.adminGroups}`
Array of Strings of context admin users	`$.roles.contextAdminUsers`	`${roles.contextAdminUsers}`
Array of Strings of context admin groups	`$.roles.contextAdminGroups`	`${roles.contextAdminGroups}`

Accessing Contextual Information

Get Information About the Environment Using the Information API

Description	Script Task Object/Property (Read Access)	JUEL Expression (Read-only)	Type
Business key of the current workflow instance	`$.info.businessKey`	`${info.businessKey}`	String

Description	Script Task Object/Property (Read Access)	JUEL Expression (Read-only)	Type
ID of the currently executed workflow definition	`$.info.workflowDefinitionId`	`${info.workflowDefinitionId}`	String
ID of the currently executed workflow instance	`$.info.workflowInstanceId`	`${info.workflowInstanceId}`	String
User who started the current workflow instance	`$.info.startedBy`	`${info.startedBy}`	String

Get Information About the User Tasks Using the Information API

All properties are read-only.

Type	Script Task Object/Property (Replace usertask1)	JUEL Expression (Replace usertask1)	Comment

String	`$.usertasks.usertask1.last.id`
	`$.usertasks.usertask1.last.createdBy`
	`$.usertasks.usertask1.last.priority`
	`$.usertasks.usertask1.last.status`
	`$.usertasks.usertask1.last.subject`
	`$.usertasks.usertask1.last.description`
	`$.usertasks.usertask1.last.processor`
		`${usertasks.usertask1.last.id}`
		`${usertasks.usertask1.last.createdBy}`
		`${usertasks.usertask1.last.priority}`
		`${usertasks.usertask1.last.status}`
		`${usertasks.usertask1.last.subject}`
		`${usertasks.usertask1.last.description}`
		`${usertasks.usertask1.last.processor}`
		`${usertasks.usertask1.last.createdAt}`
		`${usertasks.usertask1.last.dueDate}`
		`${usertasks.usertask1.last.claimedAt}`
		`${usertasks.usertask1.last.completedAt}`
			`processor`, `claimedAt`, and `completedAt` are available based on the status

Date	`$.usertasks.usertask1.last.createdAt`
	`$.usertasks.usertask1.last.dueDate`
	`$.usertasks.usertask1.last.claimedAt`
	`$.usertasks.usertask1.last.completedAt`
		Not available with the Date type, use an expression of type String instead.	• `claimedAt` and `completedAt` are available based on the status.
			• When stored into the context, dates are converted to strings.

Type	Script Task Object/Property (Replace usertask1)	JUEL Expression (Replace usertask1)	Comment
Arrays of strings	`$.usertasks.usertask1.last.recipientUsers` `$.usertasks.usertask1.last.recipientGroups`	`${usertasks.usertask1.last.recipientUsers}` `${usertasks.usertask1.last.recipientGroups}`	n.a.

3.1.10 Legacy: Enable the Workflow Editor in SAP Web IDE

You must enable the workflow editor extension to model workflows in SAP Web IDE Full-Stack.

Prerequisites

You have enabled the SAP Web IDE Full-Stack version. You must use the SAP Web IDE Full-Stack version. For more information, see Opening SAP Web IDE.

Procedure

1. Log in to the SAP Web IDE application.
2. From the left sidebar, open the *Preferences* perspective by choosing (Preferences).
3. Choose *Extensions*.
4. Enable the *Workflow Editor* extension by using the toggle.
5. Choose *Save*.
6. Reload SAP Web IDE by choosing *Refresh*.

3.1.10.1 Model Workflows in a Multitarget Application Project with SAP Web IDE

A workflow module can hold one or more workflows.

You can model workflows in a multitarget application (MTA) project in the following ways:

- MTA project containing workflow module. For more information, see MTA Project Containing Workflow Module [page 87].

- MTA project containing workflow module and custom task UI. For more information, see MTA Project Containing Workflow Module and Custom Task User Interface [page 88].
- MTA project containing workflow module and SAP Fiori launchpad. For more information, see MTA Project Containing Workflow Module and SAP Fiori Launchpad [page 92].
- MTA project containing workflow module, custom task UI, and SAP Fiori launchpad. For more information, see MTA Project Containing Workflow Module, Custom Task UI, and SAP Fiori Launchpad [page 97].
- MTA project containing workflow module and SAP Fiori launchpad with Custom Tile for Start UI. For more information, see MTA Project Containing Workflow Module and SAP Fiori Launchpad With Custom Tile for Start UI [page 100].

3.1.10.1.1 MTA Project Containing Workflow Module

You can model workflows in the workflow module.

Prerequisites

Enable the workflow editor extension to model workflows in SAP Web IDE Full-Stack. For more information, see Legacy: Enable the Workflow Editor in SAP Web IDE [page 86].

Procedure

1. Log in to the SAP Web IDE Full-Stack application.
2. From the left pane, choose </> (Development) and navigate to the *Workspace* folder.
3. Create a multitarget application (MTA) project in the SAP BTP, Cloud Foundry environment. For more information, see Create a Project from Scratch.
4. From the context menu of the MTA project folder, choose *New* > *Workflow Module*.

   > **i Note**
   >
   > You can create multiple workflow modules in the same MTA project.

5. On the *Basic Information* screen, provide a module name, then choose *Next*.
6. Provide a name and description for the workflow.
7. Choose *Finish*.

   > **i Note**
   >
   > - To create multiple workflows, you can select a workflow module or the workflow folder and choose *New* > *Workflow* from its context menu. This action creates another workflow within the selected module or the workflow folder.
   > - We recommend that you create workflows in the *workflow* folder.

8. Build and deploy the MTA project. For more information, see Build and Deploy the Workflow Module [page 75].

3.1.10.1.2 MTA Project Containing Workflow Module and Custom Task User Interface

You can model workflows with custom task UIs in the workflow module.

You can model workflows with custom task UIs in a multitarget application (MTA) project in the following ways:

- MTA project containing workflow module and custom task UI using HTML5 module. For more information, see MTA Project Containing Workflow Module and Custom Task UI Using HTML5 Module [page 88].
- MTA project containing workflow module and custom task UI using workflow forms. For more information, see MTA Project Containing Workflow Module and Custom Task UI Using Workflow Forms [page 92].

3.1.10.1.2.1 MTA Project Containing Workflow Module and Custom Task UI Using HTML5 Module

You can model workflows with custom task UIs using HTML5 module in the workflow module.

Prerequisites

Enable the workflow editor extension to model workflows in SAP Web IDE Full-Stack. For more information, see Legacy: Enable the Workflow Editor in SAP Web IDE [page 86].

Procedure

1. Log in to the SAP Web IDE Full-Stack application.
2. From the navigation pane, choose (Development) and navigate to the *Workspace* folder.
3. Create a multitarget application (MTA) project in the SAP BTP, Cloud Foundry environment. For more information, see Create a Project from Scratch.

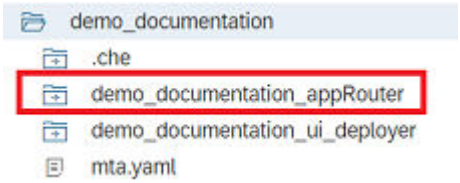
   > **Note**
   >
   > Ensure that you select *Use HTML5 Application Repository* when you create an MTA project containing UI modules.

4. Create a workflow module in the MTA project. For more information, see MTA Project Containing Workflow Module [page 87].
5. Create an HTML5 module. For more information, see Developing HTML5 Modules.

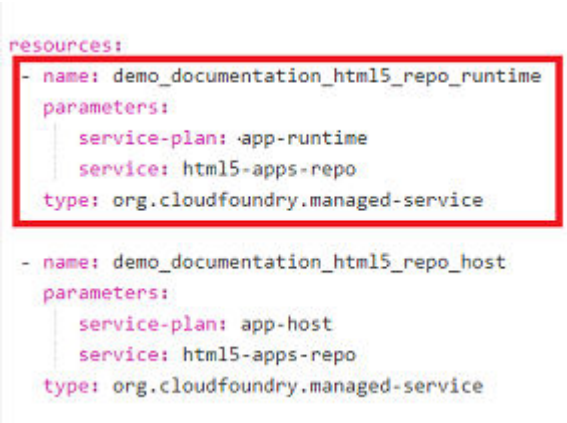
6. Select *Show hidden files* (👁) to display hidden files.

> **i Note**
>
> Ensure that you have the `deployer` module present under the created MTA project.

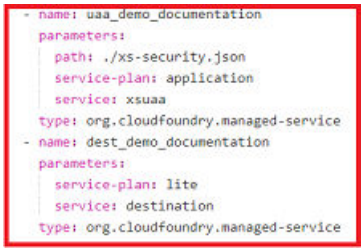
7. Delete the `<MTA ID>_appRouter` module from the MTA project.



8. In the *Code editor*, open the `mta.yaml` file under your MTA project folder.
   - For application router module:
     1. Remove `<MTA ID>_html5_repo_runtime` from the `resources` section as shown in the following image.



     2. Delete `dest_<MTA ID>` from the `resources` section as shown in the following image.



9. Delete the `xs-security.json` file from the MTA project folder.
10. Provide the route information to access the REST-based APIs from the workflow service.

> **i Note**
>
> The `workflow_rest_url` is the endpoint of the workflow service instance.

To add the route information, navigate to the MTA project, select the HTML5 module, open the `xs-app.json` file, and replace the content with the following code:

```
{
                "welcomeFile": "/index.html",
```

```
                "authenticationMethod": "route",
                "logout": {
                                "logoutEndpoint": "/do/logout"
                },
                "routes": [{
                                "source": "^/bpmworkflowruntime/(.*)$",
                                "target": "/$1",
                                "service": "com.sap.bpm.workflow",
                                "endpoint": "workflow_rest_url",
                                "authenticationType": "xsuaa"
                }, {
                                "source": "^(.*)$",
                                "target": "$1",
                                "service": "html5-apps-repo-rt",
                                "authenticationType": "xsuaa"
                }]
}
```

11. To extend the UI5 application, proceed with the following steps:
    a. Set the Task and Task Context Models [page 110]
    b. Bind a UI Element to the Task Context Model [page 111]
    c. Add Task Completion Buttons to My Inbox [page 112]
    d. (Optional) Access the User Task Data [page 114]
12. Configure the custom task UI. For more information, see Configure a Custom Task User Interface Using an HTML5 App [page 33].
13. Build and deploy your project. For more information, see Build and Deploy the Workflow Module [page 75].

Results

The page element of the `webapp/view/<view name>.view.xml` should look similar to the following:

Sample Code

```
<Page showHeader="false" showFooter="false">
    <content>
        <Text text="{/context/text}" maxLines="0" id="__text0"/>
    </content>
</Page>
```

The `init` function of `webapp/Component.js` should look similar to the following:

Note

Configure the following code by providing the **<app id>**. This is the value of the **id** in the `sap.app` section of the `manifest.json` file. To find this section, start the *Code Editor* in SAP Web IDE and open *MTA project* > *HTML5 module* > *webapp* > *manifest.json*.

Sample Code

```
init: function() {
    UIComponent.prototype.init.apply(this, arguments);
    this.setModel(models.createDeviceModel(), "device");
```

```
    var startupParameters = this.getComponentData().startupParameters;
    var taskModel = startupParameters.taskModel;
    var taskId = taskModel.getData().InstanceID;

    var contextModel = new sap.ui.model.json.JSONModel("/<app id>/
bpmworkflowruntime/v1/task-instances/" + taskId + "/context");
    contextModel.setDefaultBindingMode(sap.ui.model.BindingMode.OneWay);
    this.setModel(contextModel);

    startupParameters.inboxAPI.addAction({
        action: "Reject",
        label: "Reject"
    }, function(button) {
        this._completeTask(taskId, false);
    }, this);

    startupParameters.inboxAPI.addAction({
        action: "Approve",
        label: "Approve"
    }, function(button) {
        this._completeTask(taskId, true);
    }, this);
},
```

Below this function, there should be previously created functions:

Sample Code

```
_completeTask: function(taskId, approvalStatus) {
    var token = this._fetchToken();
    $.ajax({
        url: "/<app id>/bpmworkflowruntime/v1/task-instances/" + taskId,
        method: "PATCH",
        contentType: "application/json",
        async: false,
        data: "{\"status\": \"COMPLETED\", \"context\": {\"approved\":\"" +
approvalStatus + "\"}}",
        headers: {
            "X-CSRF-Token": token
        }
    });
    this._refreshTask(taskId);
},

_fetchToken: function() {
    var token;
    $.ajax({
        url: "/<app id>/bpmworkflowruntime/v1/xsrf-token",
        method: "GET",
        async: false,
        headers: {
            "X-CSRF-Token": "Fetch"
        },
        success: function(result, xhr, data) {
            token = data.getResponseHeader("X-CSRF-Token");
        }
    });
    return token;
},

_refreshTask: function(taskId) {
    this.getComponentData().startupParameters.inboxAPI.updateTask("NA",
taskId);
}
```

3.1.10.1.2.2 MTA Project Containing Workflow Module and Custom Task UI Using Workflow Forms

You can model workflows with custom task UIs using forms in the workflow module.

Prerequisites

Enable the workflow editor extension to model workflows in SAP Web IDE Full-Stack. For more information, see Legacy: Enable the Workflow Editor in SAP Web IDE [page 86].

Procedure

1. Log in to the SAP Web IDE Full-Stack application.
2. From the navigation pane, choose </> (Development) and navigate to the *Workspace* folder.
3. Create a multitarget application (MTA) project in the SAP BTP, Cloud Foundry environment. For more information, see Create a Project from Scratch.

   > **i Note**
   >
   > Ensure that you have not selected *Use HTML5 Application Repository* when you create an MTA project.

4. Create a workflow module in the MTA project. For more information, see MTA Project Containing Workflow Module [page 87].
5. Create a form under workflow module in the MTA project. For more information, see Configure a User Task UI Using Workflow Forms [page 38]
6. Build and deploy your project. For more information, see Build and Deploy the Workflow Module [page 75].

3.1.10.1.3 MTA Project Containing Workflow Module and SAP Fiori Launchpad

You can model workflows and create the *Workflow Definition*, *Workflow Instances*, and My Inbox tiles on SAP Fiori launchpad.

Prerequisites

Enable the workflow editor extension to model workflows in SAP Web IDE Full-Stack. For more information, see Legacy: Enable the Workflow Editor in SAP Web IDE [page 86].

Procedure

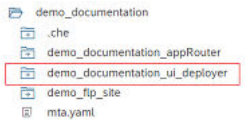
1. Log in to the SAP Web IDE Full-Stack application.
2. From the left pane, choose </> (Development) and navigate to the *Workspace* folder.
3. Create a multitarget application (MTA) project. For more information, see Create a Multitarget Application and Add Modules.

   > **i Note**
   >
   > Ensure that you select the option *Use HTML5 Application Repository* to create an application router module and a deployer module.

4. Create a workflow module in the MTA project. For more information, see MTA Project Containing Workflow Module [page 87].
5. Select *Show hidden files* (👁) to display hidden files.

   > **i Note**
   >
   > Ensure that you have the application router module and the deployer module present under the created MTA project.

6. Add an SAP Fiori launchpad module. For more information, see Add an SAP Fiori Launchpad Module.
7. Navigate to the `mta.yaml` file located under your MTA project folder and note the **ID**, which is referred as MTA ID in subsequent steps.
8. As the custom task UI is not used in this scenario, delete the `<MTA ID>_ui_deployer` module from the MTA project folder.



9. In the *Code Editor*, open the `mta.yaml` file located under your MTA project folder.
   - For SAP Fiori launchpad module:
     1. Remove dependencies of the deleted deployer module by deleting the `<MTA ID>_html5_repo_host` and `<MTA ID>_ui_deployer` service bindings from the `requires` section. Additionally, remove the resource named `<MTA ID>_html5_repo_host` from the `resources` section as shown in the following image.

```
  - name: demo_flp_site
    type: com.sap.portal.content
    path: demo_flp_site
    parameters:
      stack: cflinuxfs3
      memory: 128M
      buildpack: 'https://github.com/cloudfoundry/nodejs-buildp
    requires:
      - name: portal_resources_demo_documentation
      - name: uaa_demo_documentation
      - name: demo_documentation_html5_repo_host
      - name: demo_documentation_ui_deployer
resources:
  - name: demo_documentation_html5_repo_runtime
    parameters:
      service-plan: app-runtime
      service: html5-apps-repo
    type: org.cloudfoundry.managed-service
  - name: demo_documentation_html5_repo_host
    parameters:
      service-plan: app-host
      service: html5-apps-repo
    type: org.cloudfoundry.managed-service
  - name: portal_resources_demo_documentation
    parameters:
```

2. Add dependency to the workflow service in the `requires` section of the SAP Fiori launchpad site module.

> **i Note**
>
> - Verify that you have provided correct indentations using spaces in the `mta.yaml` file.
> - `<workflow_service_instance_name>` is the workflow service instance name created in the cockpit.
> - `<FLP module name>` is the SAP Fiori launchpad module name created in SAP Web IDE.

```
- name: <FLP module name>
  type: com.sap.portal.content
  path: <FLP module path>
  parameters:
    stack: cflinuxfs3
    memory: 128M
    buildpack: https://github.com/cloudfoundry/nodejs-buildpack/
releases/download/v1.6.21/nodejs-buildpack-v1.6.21.zip
  requires:
    - name: portal_resources_<MTA ID>
    - name: <workflow_service_instance_name>
    - name: uaa_<MTA ID>
```

3. Add dependency to the workflow service instance in the `resources` section of the SAP Fiori launchpad site module.

```
- name: <workflow_service_instance_name>
    type: org.cloudfoundry.existing-service
```

- Add `uaa` dependency:

1. Create an `xs-security.json` file present at the same level as the `mta.yaml` file in the MTA folder. Add the following code, providing a unique `xsappname` and a `role-templates` name.

```
{
  "xsappname":"<unique_xsapp_name>",
  "tenant-mode": "dedicated",
  "description": "Security profile of called application",
  "scopes": [
    {
      "name": "uaa.user",
      "description": "UAA"
    }
  ],
  "role-templates": [
    {
      "name": "<unique_role_templates_name>",
      "description": "UAA",
      "scope-references": [
        "uaa.user"
      ]
    }
  ]
}
```

2. In the `mta.yaml` file, add the following code under the `resources` section:

```
  - name: uaa_<MTA ID>
    parameters:
      path: ./xs-security.json
      service-plan: application
      service: xsuaa
    type: org.cloudfoundry.managed-service
```

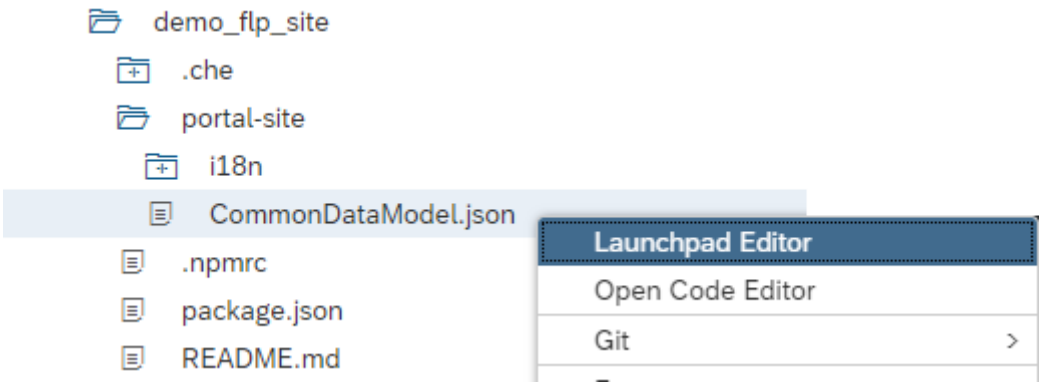
- For application router module, add the dependency to the workflow service and the uaa service under the `requires` section of the application router module.

```
- name: <MTA ID>_appRouter
  type: approuter.nodejs
  path: <MTA ID>_appRouter
  parameters:
    disk-quota: 256M
    memory: 256M
  requires:
    - name: <MTA ID>_html5_repo_runtime
    - name: portal_resources_<MTA ID>
    - name: <workflow_service_instance_name>
    - name: uaa_<MTA ID>
```

10. In the `xs-app.json` file, inside `<MTA ID>_appRouter` of the application router module, change the `authenticationMethod` to `route` by replacing the content of the file with the following code:

```
{
    "welcomeFile": "/cp.portal",
    "authenticationMethod": "route"
}
```

11. Add the tiles on the SAP Fiori launchpad.
    a. Open the `CommonDataModel.json` file located under the *portal-site* folder in the SAP Fiori launchpad site module in the *Launchpad Editor*.



b. Under *Group Tile* tab, choose ✎ to edit the *Default Group Title* and choose + to add tiles.

c. Add each tile by choosing the + icon in the top-right corner.

d. Provide the following details for *App ID* and *Intent Navigation*.

App Title	App ID	Intent Navigation
Workflow Instances	`com.sap.bpm.monitorworkflow`	`bpmworkflowmonitor-DisplayInstances`
Workflow Definition	`com.sap.bpm.monitorworkflow`	`bpmworkflowmonitor-DisplayDefinitions`
My Inbox	`cross.fnd.fiori.inbox`	`WorkflowTask-DisplayMyInbox`

e. Choose *Select* to finish.

12. Build and deploy your project. For more information, see Build and Deploy the Workflow Module [page 75].
13. Add the required roles to users. For more information, see Authorization Configuration [page 229].
14. Access the SAP Fiori launchpad with the Monitor Workflows App tile, see Access Launchpad Runtime.
    You can now see the tiles with the titles *Workflow Instances*, *Workflow Definition*, and *My Inbox* on SAP Fiori launchpad.

3.1.10.1.4 MTA Project Containing Workflow Module, Custom Task UI, and SAP Fiori Launchpad

You can model workflows with custom task UIs and create the *Workflow Definition*, *Workflow Instances*, and My Inbox tiles on SAP Fiori launchpad.

Prerequisites

Enable the workflow editor extension to model workflows in SAP Web IDE Full-Stack. For more information, see Legacy: Enable the Workflow Editor in SAP Web IDE [page 86].

Procedure

1. Log in to the SAP Web IDE Full-Stack application.
2. From the navigation pane, choose </> (Development) and navigate to the *Workspace* folder.
3. Create a multitarget application (MTA) project in the SAP BTP, Cloud Foundry environment. For more information, see Create a Project from Scratch.

   > **i Note**
   >
   > Ensure that you select *Use HTML5 Application Repository* when you create an MTA project containing UI modules.

4. Create a workflow module in the MTA project. For more information, see MTA Project Containing Workflow Module [page 87].
5. Create an HTML5 module. For more information, see Developing HTML5 Modules.
6. Select *Show hidden files* (👁) to display hidden files.

   > **i Note**
   >
   > Ensure that you have the application router module and the deployer module present under the created MTA project.

7. Provide the route information to access the REST-based APIs from the workflow service.

   > **i Note**
   >
   > The `workflow_rest_url` is the end point of the workflow service instance.

   To add the route information, navigate to the MTA project, select the HTML5 module, open the `xs-app.json` file and replace the content with the following code:

```
{
                "welcomeFile": "/index.html",
                "authenticationMethod": "route",
                "logout": {
```

```
                                "logoutEndpoint": "/do/logout"
                },
                "routes": [{
                                "source": "^/bpmworkflowruntime/(.*)$",
                                "target": "/$1",
                                "service": "com.sap.bpm.workflow",
                                "endpoint": "workflow_rest_url",
                                "authenticationType": "xsuaa"
                }, {
                                "source": "^(.*)$",
                                "target": "$1",
                                "service": "html5-apps-repo-rt",
                                "authenticationType": "xsuaa"
                }]
}
```

8. To extend the UI5 application, proceed with the following steps:
   a. Set the Task and Task Context Models [page 110]
   b. Bind a UI Element to the Task Context Model [page 111]
   c. Add Task Completion Buttons to My Inbox [page 112]
   d. (Optional) Access the User Task Data [page 114]
9. Add an SAP Fiori launchpad module. For more information, see Add an SAP Fiori Launchpad Module.
10. Navigate to the `mta.yaml` file, located under your MTA project folder and note the **ID**, which is referred as MTA ID in subsequent steps.
11. In the *Code Editor*, open the `mta.yaml` file located under your MTA project folder.
    - For SAP Fiori launchpad module:
      1. Add a dependency to the workflow service in the `requires` section of the SAP Fiori launchpad site module.

         > **i Note**
         > - Verify that you have provided correct indentations using spaces in the `mta.yaml` file.
         > - `<workflow_service_instance_name>` is the workflow service instance name created in the cockpit.
         > - `<FLP module name>` is the SAP Fiori launchpad module name created in SAP Web IDE.

         ```
         - name: <FLP module name>
           type: com.sap.portal.content
           path: <FLP module path>
           parameters:
             stack: cflinuxfs3
             memory: 128M
             buildpack: https://github.com/cloudfoundry/nodejs-buildpack/
         releases/download/v1.6.21/nodejs-buildpack-v1.6.21.zip
           requires:
             - name: portal_resources_<MTA ID>
             - name: <workflow_service_instance_name>
             - name: uaa_<MTA ID>
         ```

      2. Add dependency to the workflow service instance in the `resources` section of the SAP Fiori launchpad site module.

         ```
         - name: <workflow_service_instance_name>
             type: org.cloudfoundry.existing-service
         ```

> **Note**
>
> - As you have created an HTML5 module in your MTA project, the `uaa` dependency is already available.
> - The `dest_<MTA ID>` gets created in the `resources` section when you add an HTML5 module to your MTA project. You can delete this resource if it is not required.

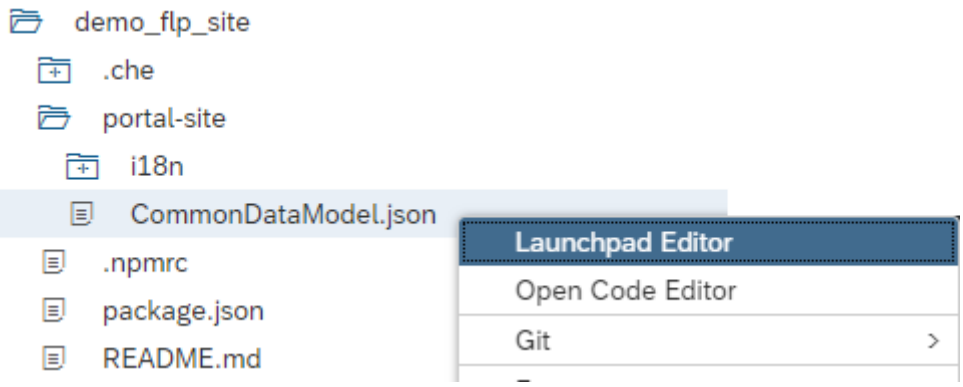
- For application router module, add a dependency to the workflow service under the `requires` section of the application router module.

```
- name: <MTA ID>_appRouter
  type: approuter.nodejs
  path: <MTA ID>_appRouter
  parameters:
    disk-quota: 256M
    memory: 256M
  requires:
    - name: <MTA ID>_html5_repo_runtime
    - name: portal_resources_<MTA ID>
    - name: <workflow_service_instance_name>
    - name: uaa_<MTA ID>
```

12. In the `xs-app.json` file, inside `<MTA ID>_appRouter` of the application router, change the `authenticationMethod` to `route` by replacing the content of the file with the following code:

```
{
    "welcomeFile": "/cp.portal",
    "authenticationMethod": "route"
}
```

13. Add the tiles on the SAP Fiori launchpad.
    a. Open the `CommonDataModel.json` file located under the *portal-site* folder in the SAP Fiori launchpad site module in the *Launchpad Editor*.



    b. On the *Group Tile* tab, edit the *Default Group Title* using the ✎ icon and choose + icon to add tiles.
    c. Add each tile by choosing the + icon in the top-right corner.
    d. Provide the following details for *App ID* and *Intent Navigation*.

App Title	App ID	Intent Navigation
Workflow Instances	`com.sap.bpm.monitorworkflow`	`bpmworkflowmonitor-DisplayInstances`
Workflow Definition	`com.sap.bpm.monitorworkflow`	`bpmworkflowmonitor-DisplayDefinitions`
My Inbox	`cross.fnd.fiori.inbox`	`WorkflowTask-DisplayMyInbox`

e. Choose *Select* to finish.

14. Build and deploy your project. For more information, see Build and Deploy the Workflow Module [page 75].
15. Add the required roles to users. For more information, see Authorization Configuration [page 229].
16. Access the SAP Fiori launchpad with the Monitor Workflows App tile, see Access Launchpad Runtime. You can now see the tiles with the titles *Workflow Instances*, *Workflow Definition*, and My Inbox on SAP Fiori launchpad.

3.1.10.1.5 MTA Project Containing Workflow Module and SAP Fiori Launchpad With Custom Tile for Start UI

You can create custom tiles on SAP Fiori launchpad.

Prerequisites

Enable the workflow editor extension to model workflows in SAP Web IDE Full-Stack. For more information, see Legacy: Enable the Workflow Editor in SAP Web IDE [page 86].

Procedure

1. Log in to the SAP Web IDE Full-Stack application.
2. From the navigation pane, choose (Development) and navigate to the *Workspace* folder.
3. Create a multitarget application (MTA) project containing workflow module. For more information, see MTA Project Containing Workflow Module [page 87].
4. Create a Start UI. For more information, see Creating a Custom Start UI [page 120].

5. Add the following code in the `sap.app` section of the `manifest.json` file located under the `webapp folder` of the HTML5 module created as part of the previous step.

```
"crossNavigation": {
            "context": {},
            "inbounds": {
                "newtile": {
                    "semanticObject": "<input>",
                    "action": "<input_action_name>",
                    "icon": "sap-icon://<icon_name>",
                    "signature": {
                        "parameters": {},
                        "additionalParameters": "<allowed/notallowed>"
                    },
                    "title": "<customTileTitle>",
                    "subTitle": "<customTileSubTitle>"
                }
            }
        }
```

6. Select *Show hidden files* (👁) to display hidden files.

> **i Note**
>
> Ensure that you have the application router module and the deployer module present under the created MTA project.

7. Add an SAP Fiori launchpad module. For more information, see Add an SAP Fiori Launchpad Module.
8. Navigate to the `mta.yaml` file located under your MTA project folder and note the **ID**, which is referred as MTA ID in subsequent steps.
9. In the *Code Editor*, open the `mta.yaml` file located under your MTA project folder.
   - For SAP Fiori launchpad module:
     1. Add a dependency to the workflow service in the `requires` section of the SAP Fiori launchpad site module.

        > **i Note**
        >
        > - Verify that you have provided correct indentations using spaces in the `mta.yaml` file.
        > - `<workflow_service_instance_name>` is the workflow service instance name created in the cockpit.
        > - `<FLP module name>` is the SAP Fiori launchpad module name created in SAP Web IDE.

        ```
        - name: <FLP module name>
          type: com.sap.portal.content
          path: <FLP module path>
          parameters:
            stack: cflinuxfs3
            memory: 128M
            buildpack: https://github.com/cloudfoundry/nodejs-buildpack/
        releases/download/v1.6.21/nodejs-buildpack-v1.6.21.zip
          requires:
            - name: portal_resources_<MTA ID>
            - name: <workflow_service_instance_name>
            - name: uaa_<MTA ID>
        ```

     2. If there is no dependency added to the workflow service instance, add the following code in the `resources` section:

```
- name: <workflow_service_instance_name>
    type: org.cloudfoundry.existing-service
```

> **i Note**
>
> - As you have created an HTML5 module in your MTA project, the `uaa` dependency is already available.
> - The `dest_<MTA ID>` gets created in the `resources` section when you add an HTML5 module to your MTA project. You can delete this resource if it is not required.

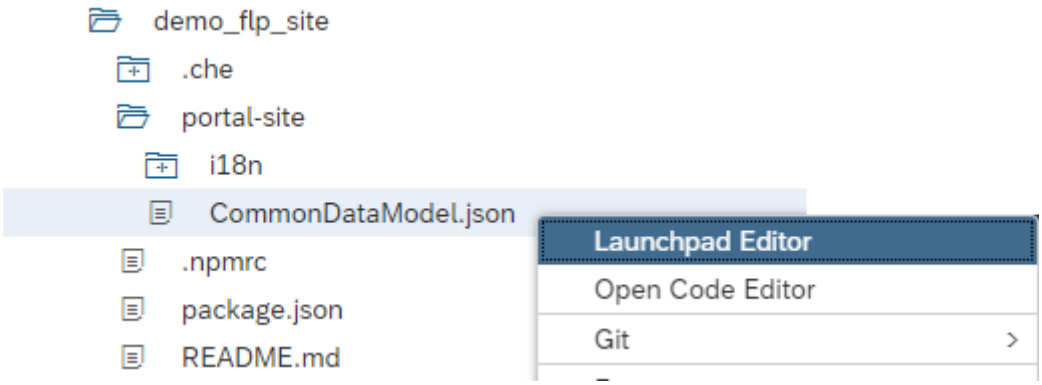
- For application router module:
  1. Add a dependency to the workflow service and the uaa service under the `requires` section of the application router.

```
- name: <MTA ID>_appRouter
  type: approuter.nodejs
  path: <MTA ID>_appRouter
  parameters:
    disk-quota: 256M
    memory: 256M
  requires:
    - name: <MTA ID>_html5_repo_runtime
    - name: portal_resources_<MTA ID>
    - name: <workflow_service_instance_name>
    - name: uaa_<MTA ID>
```

  2. In the `xs-app.json` file inside `<MTA ID>_appRouter` of the application router module, change the `authenticationMethod` to `route` by replacing the content of the file with the following code:

```
{
    "welcomeFile": "/cp.portal",
    "authenticationMethod": "route"
}
```

10. Add the tiles on the SAP Fiori launchpad.
    a. Open the `CommonDataModel.json` file located under the *portal-site* folder in the SAP Fiori launchpad site module in the *Launchpad Editor*.



    b. On the *Group Tile* tab, edit the *Default Group Title* using the ✎ icon and choose + icon to add tiles.
    c. Choose the + icon to add a new tile in the top-right corner.
    d. Select your custom tile.
    e. Choose *Select* to finish.
11. Build and deploy your project. For more information, see Build and Deploy the Workflow Module [page 75].

12. Add the required roles to users. For more information, see Authorization Configuration [page 229].
13. Access the SAP Fiori launchpad with custom tile, see Access Launchpad Runtime.
    You can now see the custom tile on SAP Fiori launchpad.

3.1.10.1.6 Modify the Multitarget Application

You need to adapt the multitarget application (MTA).

Prerequisites

You've created an MTA project with a workflow module. See Model Workflows in a Multitarget Application Project with SAP Web IDE [page 86].

Procedure

1. Modify the multitarget application (MTA) by providing either of the following resources types and parameters in the MTA deployment descriptor (`mta.yaml`).
   - To use an existing instance of workflow service, define it by using `org.cloudfoundry.existing-service` resource type along with the existing resource name.

     You can find the existing instance name of workflow service by navigating to *Services* > *Service Instances* in the cockpit.

     Sample Code

     ```
     resources:
       - name: <existing_workflow_service_instance_name>
         type: org.cloudfoundry.existing-service
     ```

   - If there's no workflow service instance created, create a new workflow service instance. Define it using the `org.cloudfoundry.managed-service` resource type along with the resource name provided by the user.

     Sample Code

     ```
     resources:
       - name: <workflow_service_instance_name>
         type: org.cloudfoundry.managed-service
         parameters:
           service-plan: standard
           service: workflow
     ```

   For more information about resources types and parameters, see MTA Module Types, Resource Types, and Parameters for Applications in the Cloud Foundry Environment.

2. Add a dependency to the workflow service resource in the `requires` section of the workflow module.

> Sample Code

```
requires:
  - name: <workflow_service_instance_name>
    parameters:
      content-target: true
```

3.1.10.2 Migrate Workflow Project to a Multitarget Application Project with SAP Web IDE

You can migrate your workflow project to a multitarget application (MTA) project for your existing workflows to be used in the SAP BTP, Cloud Foundry environment environment.

Prerequisites

Create a multitarget application project and a workflow module. For more information, see Model Workflows in a Multitarget Application Project with SAP Web IDE [page 86].

> **i Note**
>
> Under the workflow module, delete all the folders except the `.che` folder.

Procedure

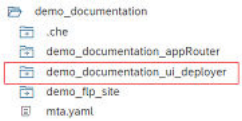
1. If your workflow project contains SAP UI5 content, create separate HTML5 modules and migrate them separately. For more information, see Developing HTML5 Modules.
2. To migrate content other than SAP UI5 content from a workflow project, follow the steps below:
   a. In the workflow project, select all folders except the `.che` folder.
   b. Right-click and select *Edit* > *Copy* from the context menu.
   c. Navigate to workflow module.
   d. Right-click and select *Edit* > *Paste* from the context menu.
3. If you have user tasks with custom task UIs using an HTML5 module in your workflow, perform the following steps to reconfigure the custom task user interface:
   a. In your workflow, select the user task.
   b. From the *User Task Properties* area, choose the *Details* tab.
   c. In the *Recipients* tab, replace the value of *Users* from ID to email address.
   d. From the *User Task Properties* area, choose the *User Interface* tab.
   e. If the *Type* is *SAPUI5 Component*, then under *HTML5 App Name*, choose *Select*.

f. In the *Choose User Interface* window, choose the MTA project that includes the SAPUI5 component.
g. Choose *SAPUI5 Component Path*.
h. Choose *OK*.

For more information, see Configure a Custom Task User Interface Using an HTML5 App [page 33]

> **i Note**
>
> If your MTA project contains only forms and you have not selected the *Use HTML5 Application Repository* checkbox while creating the MTA project, skip **Step 4**.

4. If your MTA project doesn't have an HTML5 module, perform the following steps.
   a. Delete the `<MTA ID>_ui_deployer` module if the dependency is already available.



   b. Open the `mta.yaml` file under your MTA project in *Code Editor*. Remove the resource with the name `<MTA ID>_html5_repo_host` under the `resources` section if the dependency is already available, as shown in the following image.

```
resources:
  - name: demo_documentation_html5_repo_runtime
    parameters:
      service-plan: app-runtime
      service: html5-apps-repo
    type: org.cloudfoundry.managed-service
  - name: demo_documentation_html5_repo_host
    parameters:
      service-plan: app-host
      service: html5-apps-repo
    type: org.cloudfoundry.managed-service
  - name: portal_resources_demo_documentation
    parameters:
```

5. If you have user tasks with custom task UIs using workflow forms, perform the following steps to reconfigure the user task UI:
   a. In your workflow, select the user task.
   b. From the *User Task Properties* area, choose the *User Interface* tab.
   c. If the *Type* is *Form*, then under *Form Details* section, choose *Select*.
   d. On the *Select Form* dialog, choose the required form.

Field	Description
Project Name	Name of the project and workflow module. You cannot change this information.
File Name	Name of the form.

Field	Description
Form Revision	Revision number of the form. For more information, see Versioning Forms [page 147].

For more information, see Configure a User Task UI Using Workflow Forms [page 38].

6. Save your workflow.

   If you do not need to modify the project further, proceed to Build and Deploy the Workflow Module [page 75]

3.1.10.3 Open Workflow Files in the Workflow Editor with SAP Web IDE

Open existing workflow files in the workflow editor to view or modify them.

Procedure

1. Log in to the SAP Web IDE application.
2. From the navigation pane, choose </> (Development), and navigate to the *Workspace* folder.
3. Select the multitarget application (MTA) project, and navigate to the workflow module.
4. Navigate to the *workflows* folder in the context menu of your workflow file, and choose *Workflow Editor*.

3.2 Creating User Interfaces

With SAP Workflow service, you can create user interfaces for workflows.

With the user interfaces, end users can access their workflow tasks in their inboxes and start workflows. You have the following options:

- Creating Custom UIs
  With the REST-based APIs of the workflow service, you can access the workflow service runtime. On top of these APIs, you can develop scenario-specific user interfaces (UIs) for user tasks or workflow start UIs.
  - Creating a Custom Start UI [page 120]
    With SAPUI5, this option gives you a high control of the UI.
  - Creating a Custom Task UI [page 107]
    This option enables you to create a simple straightforward UI using predefined elements.
- Creating a Workflow Form [page 132]
  This option enables you to declaratively model straightforward UIs using predefined elements for simple task UIs as well as start UIs.

Related Information

Using Workflow APIs [page 151]

3.2.1 Creating a Custom Task UI

You can use the custom task user interface as a UI of a user task in your workflow definition.

Prerequisites

In SAP Business Application Studio, you have created a dev space with the following extensions:

- *SAP Predefined Extension MTA Tools* available, for example, with the application kind *SAP Fiori*.
- *Additional SAP Extension Workflow Management*

Procedure

1. In SAP Business Application Studio, open your workspace, for example, the *projects* folder.
2. To create a Fiori project with an HTML5 module, that contains your custom task UI, choose *File* > *New Project from Template*.

   Select the following configuration settings, and choose *Next* to move to the next setting:

Field	Value
Select Template and Target Location	*SAP Fiori Freestyle Project*
Target Running Environment	*Cloud Foundry*
Template	*SAPUI5 Application*
Project Name	Enter text
HTML5 application runtime	*Managed Approuter*
Enter a unique name for the business solution of the project	Enter the business solution name of your application.
Enter an HTML5 module name	Leave prefilled entry.

Field	Value
Do you want to add authentication?	*Yes*
Enter a namespace	Leave prefilled entry.
Do you want to enable Karma tests?	Leave prefilled entry.
Enter a view name	Leave prefilled entry.
Do you want to add a data service?	*No*

3. Change the scope of the destination module.
   a. Navigate to your newly created Fiori project.
   b. Open the `mta.yaml` file in your newly created project.
   c. Locate the destination module with the `destination-content` suffix.
   d. Change the scope of the destination by replacing "instance:" with "subaccount:" for the content parameter of the module. A resulting module configuration may look as the example below.

Sample Code

```
modules:
- name: my.project-destination-content
  type: com.sap.application.content
  requires:
  - name: my.project-destination-content
    parameters:
      content-target: true
  - name: my.project_html_repo_host
    parameters:
      service-key:
        name: my.project_html_repo_host-key
  - name: uaa_my.project
    parameters:
      service-key:
        name: uaa_my.project-key
  parameters:
    content:
      subaccount:
        destinations:
        - Name: my_business_solution_name_my_project_html_repo_host
          ServiceInstanceName: my.project-html5-app-host-service
          ServiceKeyName: my.project_html_repo_host-key
          sap.cloud.service: my.business.solution.name
        - Authentication: OAuth2UserTokenExchange
          Name: my_business_solution_name_uaa_my_project
          ServiceInstanceName: my.project-xsuaa-service
          ServiceKeyName: uaa_my.project-key
          sap.cloud.service: my.business.solution.name
        existing_destinations_policy: ignore
  build-parameters:
    no-source: true
```

> **Note**
>
> Do not modify the indents of the lines in the `mta.yaml` file.

4. Provide the route information to access the REST-based APIs from the workflow service.
   a. Navigate to your newly created Fiori project.
   b. In the *HTML5Module* folder, open the `xs-app.json` file.
   c. Add a `bpmworkflowruntime` route information as the first route in the routes configuration.

      The resulting configuration looks as follows:

      > **Sample Code**

      ```
      {
        "welcomeFile": "/index.html",
        "authenticationMethod": "route",
        "logout": {
          "logoutEndpoint": "/do/logout"
        },
        "routes": [
          {
              "source": "^/bpmworkflowruntime/(.*)$",
              "target": "/$1",
              "service": "com.sap.bpm.workflow",
              "endpoint": "workflow_rest_url",
              "authenticationType": "xsuaa"
          },
          {
            "source": "^(.*)$",
            "target": "$1",
            "service": "html5-apps-repo-rt",
            "authenticationType": "xsuaa"
          }
        ]
      }
      ```

      > **Note**
      >
      > You must add the `bpmworkflowruntime` route as the first route in the routes section.
      >
      > Do not change the `workflow_rest_url` endpoint of the workflow service instance.

5. To extend the UI5 application, proceed with the following steps:
   a. Set the Task and Task Context Models [page 110]
   b. Bind a UI Element to the Task Context Model [page 111]
   c. Add Task Completion Buttons to My Inbox [page 112]
   d. (Optional) Access the User Task Data [page 114]

Related Information

Configure User Tasks [page 29]

3.2.1.1 Set the Task and Task Context Models

To read the task and task context model, add the following functions to your `webapp/Component.js` file.

Procedure

1. In the *HTML5Module* folder, open *webapp* > *Component.js*.
2. Copy and paste the following functions into the file.

Sample Code

```
setTaskModels: function () {
    // set the task model
    var startupParameters = this.getComponentData().startupParameters;
    this.setModel(startupParameters.taskModel, "task");

    // set the task context model
    var taskContextModel = new
sap.ui.model.json.JSONModel(this._getTaskInstancesBaseURL() + "/context");
    this.setModel(taskContextModel, "context");
},

_getTaskInstancesBaseURL: function () {
    return this._getWorkflowRuntimeBaseURL() + "/task-instances/" +
this.getTaskInstanceID();
},

_getWorkflowRuntimeBaseURL: function() {
    var appId = this.getManifestEntry("/sap.app/id");
    var appPath = appId.replaceAll(".", "/");
    var appModulePath = jQuery.sap.getModulePath(appPath);
    return appModulePath + "/bpmworkflowruntime/v1";
},

getTaskInstanceID: function() {
    return this.getModel("task").getData().InstanceID;
}
```

Then, call the `setTaskModels` function in the component's `init` function:

Sample Code

```
init: function () {
    // call the base component's init function
    UIComponent.prototype.init.apply(this, arguments);

    // enable routing
    this.getRouter().initialize();

    // set the device model - Test test
    this.setModel(models.createDeviceModel(), "device");

    this.setTaskModels();
}
```

Results

After the models are set, the task data and task context data are available in the "task", respective "context" JSON models. You can use it for data binding.

3.2.1.2 Bind a UI Element to the Task Context Model

To display and change a field of the task context on the custom task UI, add a text element to view.xml file and bind it to an existing text property of the context JSON model.

Prerequisites

You have set the task models, see Set the Task and Task Context Models [page 110].

Procedure

1. In the *HTML5Module* folder, open *webapp* > *Component.js* > *view* > *<view name>*.
2. Copy the name of your controller and so you can paste it back in later.
3. Replace the generated page with the following code snippet.

   The resulting view XML looks as follows:

   Sample Code

   ```
   <mvc:View controllerName="<controller name>" xmlns:mvc="sap.ui.core.mvc"
   displayBlock="true" xmlns="sap.m">
       <App id="app">
           <pages>
               <Page showHeader="false" showFooter="false">
                   <content>
                       <Input value="{context>/text}" id="__input0"/>
                   </content>
               </Page>
           </pages>
       </App>
   </mvc:View>
   ```

   Note

   The view to be rendered in SAP Fiori My Inbox must not contain a Shell control. Delete the Shell control if necessary.

4. Replace the placeholder for the controller name with the one you copied from the generated file.

Results

After building and deploying the custom task UI, see Build and Deploy the Workflow Module [page 75], you can already use it for a user task of your workflow definition, see Configure User Tasks [page 29]. However, the user task cannot be completed by an end user. Therefore, a changed field value is not yet written back to the task context. For this task, you must add completion buttons, see Add Task Completion Buttons to My Inbox [page 112].

3.2.1.3 Add Task Completion Buttons to My Inbox

In My Inbox, end users can complete tasks using task completion buttons.

Prerequisites

The task models are set. See Set the Task and Task Context Models [page 110].

Procedure

1. In the *HTML5Module* folder, open *webapp* > *Component.js*.
2. To enable a convenient access to the My Inbox UI Integration API, add the following function to your `Component.js` file.

   Sample Code

   ```
   getInboxAPI: function () {
       var startupParameters = this.getComponentData().startupParameters;
       return startupParameters.inboxAPI;
   }
   ```

3. Prepare task completion.

   A task is completed by fetching an XSRF token and calling the task completion REST API, see Using Workflow APIs [page 151]. The actual task outcome can be added to the task context during completion (here property approved). The outcome is available for usage within your workflow definition, for example, in a gateway.

   To enable the task completion, add the following functions to your `Component.js` file:

   Sample Code

   ```
   completeTask: function (approvalStatus) {
       this.getModel("context").setProperty("/approved", approvalStatus);
       this._patchTaskInstance();
       this._refreshTaskList();
   },
   ```

```
_patchTaskInstance: function () {
    var data = {
        status: "COMPLETED",
        context: this.getModel("context").getData()
    }

    jQuery.ajax({
        url: this._getTaskInstancesBaseURL(),
        method: "PATCH",
        contentType: "application/json",
        async: false,
        data: JSON.stringify(data),
        headers: {
            "X-CSRF-Token": this._fetchToken()
        }
    });
},

_fetchToken: function () {
    var fetchedToken;

    jQuery.ajax({
        url: this._getWorkflowRuntimeBaseURL() + "/xsrf-token",
        method: "GET",
        async: false,
        headers: {
            "X-CSRF-Token": "Fetch"
        },
        success(result, xhr, data) {
            fetchedToken = data.getResponseHeader("X-CSRF-Token");
        }
    });
    return fetchedToken;
},

_refreshTaskList: function () {
    this.getInboxAPI().updateTask("NA", this.getTaskInstanceID());
}
```

4. Add task completion buttons using the My Inbox UI Integration API.

   To add Approve and Reject buttons, add the following code in the component's `init` function after the `setTaskModels` function (see Set the Task and Task Context Models [page 110]).

   Sample Code

```
init: function () {
    ...
    this.setTaskModels();

    this.getInboxAPI().addAction({
        action: "APPROVE",
        label: "Approve",
        type: "accept" // (Optional property) Define for positive
appearance
    }, function () {
        this.completeTask(true);
    }, this);

    this.getInboxAPI().addAction({
        action: "REJECT",
        label: "Reject",
        type: "reject" // (Optional property) Define for negative
appearance
    }, function () {
```

```
            this.completeTask(false);
        }, this);
        ...
    }
```

> **Note**
>
> - The previously created function `completeTask` is called in both actions but with different approval status.
> - In case you haven't specified the additional type parameter, the custom action button appears with a default appearance.
> - You can add any additional completion buttons. However, make sure that the action's ID is unique across all actions.

Results

After building and deploying the custom task UI, a task can be approved or rejected using the task completion buttons in the My Inbox. For more information about deploying, see Build and Deploy the Workflow Module [page 75]

3.2.1.4 Access the User Task Data

The My Inbox provides additional data on custom task UIs, for example, the task model.

Context

The My Inbox passes the additional data to a custom task UI through the UI5 component's startup parameters.

Access the Task Model

The task model contains data related to user tasks.

Context

The `taskModel` is of type `sap.ui.model.json.JSONModel` and contains the following task properties:

- `SAP__Origin`

- `InstanceID`
- `TaskDefinitionID`
- `TaskDefinitionName`
- `TaskTitle`
- `Priority`
- `PriorityText`
- `Status`
- `StatusText`
- `CreatedBy`
- `CreatedOn`
- `Processor`

`PriorityText` and **`StatusText`** contain translated texts that are specific to the *My Inbox* user's locale.

Procedure

1. In the *HTML5Module* folder, open *webapp* > *Component.js*.
2. To access the task model and retrieve the task properties within your custom task UI, add the following function to your `Component.js` file:

   Sample Code

   ```
   init: function () {
       ...
       var startupParameters = this.getComponentData().startupParameters;
       this.setModel(startupParameters.taskModel, "task");
       ...
   }
   ```

   See Set the Task and Task Context Models [page 110].

Access the My Inbox Integration UI API

You can add task completion buttons in the My Inbox, control the visibility of the footer bar or the back navigation button, and more with the My Inbox UI Integration API.

Context

For a complete list, see My Inbox UI Integration API Reference.

Procedure

1. In the *HTML5Module* folder, open *webapp* > *Component.js*.
2. To enable a convenient access to the My Inbox UI Integration API, add the following function to your `Component.js` file.

   Sample Code

   ```
   getInboxAPI: function () {
       var startupParameters = this.getComponentData().startupParameters;
       return startupParameters.inboxAPI;
   }
   ```

   The My Inbox Integration UI API provides additional instance-specific data and actions for tasks. See Add Task Completion Buttons to My Inbox [page 112].
3. Call the function of interest in the `init` function of the `Component.js` file.

   You can add its result to a JSON model, for example, the task model, to access it in your custom task UI.

   Sample Code

   ```
   init: function () {
       ...
       this.getInboxAPI().getDescription("NA",
   this.getTaskInstanceID()).done(function(data) {
           this.getModel("task").setProperty("/Description", data.Description)
       }.bind(this));
       ...
   }
   ```

Access the Query Parameters

Context

Query parameters that are set for a custom task UI can also be accessed through the startup parameters.

Procedure

1. In the *HTML5Module* folder, open *webapp* > *Component.js*.
2. To access the query parameters and retrieve the parameters of a user task, add the following lines to the `init` function of your `Component.js` file:

   Sample Code

   ```
   init: function () {
       ...
   ```

```
    var startupParameters = this.getComponentData().startupParameters;
    var queryParameters = startupParameters.oParameters.oQueryParameters
    ...
}
```

3.2.1.4.1 My Inbox UI Integration API Reference

You can use a set of APIs to integrate your task application with *My Inbox*.

addAction

Adds an action button to the My Inbox footer.

Parameters

Name	Type	Description
action	object	A JSON object specifying action details. Properties: • action : string • label : string • type : string (either Accept or Reject)
actionEventHandler	function	The function to be called when the event occurs.
listener?	object	Context object to call the event handler.

Return Value

success : A boolean representing successful addition of the button to the footer.

disableAction

Disable single custom action button in *My Inbox* button bar.

Parameters

Name	Type	Description
action	string	Name of the action to be disabled

Return Value

success : A boolean representing successful disabling of the button in the footer.

disableAllActions

Disable all custom action buttons in *My Inbox* button bar.

Return Value

success : A boolean representing successful disabling of all custom buttons in the footer.

enableAction

Enable single custom action button in *My Inbox* button bar.

Parameters

Name	Type	Description
action	string	Name of the action to be enabled

Return Value

success : A boolean representing successful enabling of the button in the footer.

enableAllActions

Enable all custom action buttons in *My Inbox* button bar.

Return Value

success : A boolean representing successful enabling of all custom buttons in the footer.

getDescription

Retrieves the task description and returns a promise that is resolved when the task description is retrieved.

Parameters

Name	Type	Description
`SAPOrigin`	string	Value for the parameter SAP__Origin for the specific task
`taskInstanceId`	string	Value for the parameter InstanceId for the specific task

Return Value

Promise: A promise that is resolved when the task description is retrieved. It is rejected with an error if the parameters SAPOrigin or taskInstanceId are passed with empty value or if the task description could not be retrieved (due to network issues).

removeAction

Removes an action added previously by the integrated application.

Parameters

Name	Type	Description
`actionName`	string	Name of the action to be removed

Return Value

success : A boolean representing successful removal of the button from the footer

setShowFooter

Shows or hides footer of the page.

Parameters

Name	Type	Description
`showFooter`	boolean	Flag representing whether to show or hide footer in the page. The default value is false.

setShowNavButton

Shows or hides navigation button in header of the page.

Parameters

Name	Type	Description
`showNavButton`	boolean	Flag representing whether to show or hide navigation button in header of the page. The default value is false.
`navEventHandler?`	function	Function to be called when navigation button is clicked.

updateTask

Updates the task in the master task list and returns a promise that is resolved when the task list is updated.

Parameters

Name	Type	Description
`SAPOrigin`	string	Value for the parameter SAP__Origin for the specific task
`taskInstanceId`	string	Value for the parameter InstanceId for the specific task

Return Value

Promise: A promise that is resolved when the task list is updated. It is rejected with an error if the parameters SAPOrigin or taskInstanceId are passed with empty value or if the task list could not be updated (due to network issues).

3.2.2 Creating a Custom Start UI

Create a sample UI for starting workflow instances.

Prerequisites

In SAP Business Application Studio, you have created a dev space with the following extensions:

- *SAP Predefined Extension MTA Tools* available, for example, with the application kind SAP Fiori.
- *Additional SAP Extension Workflow Management*

Context

The use case here is as follows. There is a particular workflow definition deployed into the workflow service runtime. A user interface is needed which would allow the end users to start the instances of the corresponding workflow. In addition, the users must be able to specify some arbitrary values that will be used in the contexts of the started instances.

You can either define the start UI using an HTML5 application (custom start UI) or using a start form (see Creating a Workflow Form [page 132]). To build a custom start UI, do the following:

Procedure

1. In SAP Business Application Studio, open your workspace, for example, the *projects* folder.
2. To create a Fiori project with an HTML5 module, that contains your custom task UI, choose *File* > *New Project from Template*.

Select the following configuration settings, and choose *Next* to move to the next setting:

Field	Value
Select Template and Target Location	*SAP Fiori Freestyle Project*
Target Running Environment	*Cloud Foundry*
Template	*SAPUI5 Application*
Project Name	Enter text
HTML5 application runtime	*Managed Approuter*
Enter a unique name for the business solution of the project	Enter the business solution name of your application.
Enter an HTML5 module name	Leave prefilled entry.
Do you want to add authentication?	*Yes*
Enter a namespace	Leave prefilled entry.
Do you want to enable Karma tests?	Leave prefilled entry.
Enter a view name	Leave prefilled entry.
Do you want to add a data service?	*No*

3. Change the scope of the destination module.
   a. Navigate to your newly created Fiori project.
   b. Open the `mta.yaml` file in your newly created project.
   c. Locate the destination module with the `destination-content` suffix.
   d. Change the scope of the destination by replacing "instance:" with "subaccount:" for the content parameter of the module. A resulting module configuration may look as the example below.

Sample Code

```
modules:
- name: my.project-destination-content
  type: com.sap.application.content
  requires:
  - name: my.project-destination-content
    parameters:
      content-target: true
  - name: my.project_html_repo_host
    parameters:
      service-key:
        name: my.project_html_repo_host-key
  - name: uaa_my.project
    parameters:
```

```
      service-key:
        name: uaa_my.project-key
  parameters:
    content:
      subaccount:
        destinations:
        - Name: my_business_solution_name_my_project_html_repo_host
          ServiceInstanceName: my.project-html5-app-host-service
          ServiceKeyName: my.project_html_repo_host-key
          sap.cloud.service: my.business.solution.name
        - Authentication: OAuth2UserTokenExchange
          Name: my_business_solution_name_uaa_my_project
          ServiceInstanceName: my.project-xsuaa-service
          ServiceKeyName: uaa_my.project-key
          sap.cloud.service: my.business.solution.name
        existing_destinations_policy: ignore
  build-parameters:
    no-source: true
```

> **i Note**
>
> Do not modify the indents of the lines in the `mta.yaml` file.

4. Provide the route information to access the REST-based APIs from the workflow service.
   a. Navigate to your newly created Fiori project.
   b. In the *HTML5Module* folder, open the `xs-app.json` file.
   c. Add a `bpmworkflowruntime` route information as the first route in the routes configuration.

      The resulting configuration looks as follows:

> **Sample Code**

```
{
  "welcomeFile": "/index.html",
  "authenticationMethod": "route",
  "logout": {
    "logoutEndpoint": "/do/logout"
  },
  "routes": [
    {
        "source": "^/bpmworkflowruntime/(.*)$",
        "target": "/$1",
        "service": "com.sap.bpm.workflow",
        "endpoint": "workflow_rest_url",
        "authenticationType": "xsuaa"
    },
    {
      "source": "^(.*)$",
      "target": "$1",
      "service": "html5-apps-repo-rt",
      "authenticationType": "xsuaa"
    }
  ]
}
```

> **i Note**
>
> You must add the `bpmworkflowruntime` route as the first route in the routes section.
>
> Do not change the `workflow_rest_url` endpoint of the workflow service instance.

5. The generated UI5 application contains a view and a controller by default. To extend the UI5 application, proceed with the following steps:
   a. Implement View Model Instantiation [page 123]
   b. Implement Workflow Instance Start [page 124]
   c. Extend the View [page 125]

Results

After building and deploying the custom start UI, you can configure it as a tile in the SAP Fiori Launchpad. See Create Custom Start UI Tiles on the Central Launchpad [page 211]. For more information about deploying, see Build and Deploy the Workflow Module [page 75].

.

3.2.2.1 Implement View Model Instantiation

The controller initializes a view model that is assigned to the view.

Context

It is used to keep user provided inputs. In this example, the model is represented by an object with two properties:

- The `initialContext` field refers to the input of the user that is used as an initial workflow instance context while starting.
- The `apiResponse` field refers to the string representation of the response of the workflow start request.

All properties have assigned default values that are displayed in the UI.

Procedure

1. In the *HTML5Module* folder, open the controller JS file created in the *webapp/controller* folder.
2. Substitute the `onInit` function with the following code snippet:

Sample Code

```
onInit: function () {
  this.getView().setModel(new sap.ui.model.json.JSONModel({
    initialContext: JSON.stringify({ someProperty: "some value" }, null,
4),
    apiResponse: ""
   }));
}
```

3.2.2.2 Implement Workflow Instance Start

The workflow is started using the corresponding HTTP call to the workflow service REST API.

Context

For more information, see Using Workflow APIs [page 151]. In this example, the input of the user is used the initial context of the workflow instance. In addition, the response of the call is assigned to the corresponding property in the view model.

Procedure

1. In the *HTML5Module* folder, open the controller JS file created in the *webapp/controller* folder.
2. Copy and paste the following functions into the file.

Sample Code

```
startWorkflowInstance: function () {
  var model = this.getView().getModel();
  var definitionId = "my.workflow.definition.id"; //Change to your
workflow definition ID
  var initialContext = model.getProperty("/initialContext");
  var data = {
    definitionId: definitionId,
    context: JSON.parse(initialContext)
  };
  $.ajax({
    url: this._getWorkflowRuntimeBaseURL() + "/workflow-instances",
    method: "POST",
    async: false,
    contentType: "application/json",
    headers: {
      "X-CSRF-Token": this._fetchToken()
    },
    data: JSON.stringify(data),
    success: function (result, xhr, data) {
      model.setProperty("/apiResponse", JSON.stringify(result, null, 4));
    },
    error: function (request, status, error) {
      var response = JSON.parse(request.responseText);
      model.setProperty("/apiResponse", JSON.stringify(response, null, 4));
    }
  });
},
_fetchToken: function () {
  var fetchedToken;
  jQuery.ajax({
    url: this._getWorkflowRuntimeBaseURL() + "/xsrf-token",
    method: "GET",
```

```
    async: false,
    headers: {
      "X-CSRF-Token": "Fetch"
    },
    success(result, xhr, data) {
      fetchedToken = data.getResponseHeader("X-CSRF-Token");
    }
  });
  return fetchedToken;
},
_getWorkflowRuntimeBaseURL: function () {
  var appId = this.getOwnerComponent().getManifestEntry("/sap.app/id");
  var appPath = appId.replaceAll(".", "/");
  var appModulePath = jQuery.sap.getModulePath(appPath);
  return appModulePath + "/bpmworkflowruntime/v1";
}
```

3. Substitute the `my.workflow.definition.id` string with the ID of the deployed workflow definition of interest.

3.2.2.3 Extend the View

Context

The view contains two text areas that are bound to the view model and a button that is assigned to the `startWorkflowInstance` controller function. With this button, users start a workflow instance. The value of the first text area (JSON) is used as the initial workflow context. The response of the workflow start request is printed out in the second text area.

Procedure

1. In the *HTML5Module* folder, open the XML file created in the *webapp/view* folder of your application.
2. Substitute the existing page with the following code:

Sample Code

```
<Page title="Workflow Start UI">
  <content>
  <VBox width="100%" direction="Column" class="sapUiTinyMargin">
    <items>
      <Text text="Initial Workflow Context:"/>
      <TextArea width="100%" value="{/initialContext}" rows="8" />
      <Button text="Start Workflow" press="startWorkflowInstance"/>
      <Text text="API Response:" class="sapUiSmallMarginTop"/>
      <TextArea width="100%" value="{/apiResponse}" rows="14"/>
    </items>
  </VBox>
  </content>
</Page>
```

3.2.2.4 Legacy: Creating a Custom Start UI

Create a sample UI for starting workflow instances.

Prerequisites

A workflow definition (for which the start UI is developed) is deployed into the workflow service runtime.

Context

The use case here is as follows. There is a particular workflow definition deployed into the workflow service runtime. A user interface is needed which would allow the end users to start the instances of the corresponding workflow. In addition, the users must be able to specify some arbitrary values that will be used in the contexts of the started instances.

You can either define the start UI using an existing SAPUI5 component or using a start form (see Creating a Workflow Form [page 132]). To build a custom start UI, do the following:

Procedure

1. Create an HTML5 Module for the Start UI [page 127].
2. Define the Route [page 127]
3. Extend the View [page 128]
4. Extend the Controller [page 128]

Related Information

SAP Web IDE Full-Stack Documentation

3.2.2.4.1 Create an HTML5 Module for the Start UI

You create your HTML5 app using standard SAP Business Technology Platform (SAP BTP) procedures.

Procedure

Create an HTML5 module.

- When using SAP Web IDE, create an HTML5 module. For more information, see Developing HTML5 Modules.
- When using SAP Business Application Studio, you can create a new SAP Fiori Project with an HTML5 module or create an HTML5 module in your existing project. For more information, see Create an SAP Fiori Project.

3.2.2.4.2 Define the Route

You have to define the route for the workflow service in the application configuration file.

Procedure

Provide the route information to access the REST-based APIs from the workflow service.

To add the route information, navigate to the MTA project, select the HTML5 module, open the `xs-app.json` file and replace the content with the following code :

```
{
                "welcomeFile": "/index.html",
                "authenticationMethod": "route",
                "logout": {
                                "logoutEndpoint": "/do/logout"
                },
                "routes": [{
                                "source": "^/bpmworkflowruntime/(.*)$",
                                "target": "/$1",
                                "service": "com.sap.bpm.workflow",
                                "endpoint": "workflow_rest_url",
                                "authenticationType": "xsuaa"
                }, {
                                "source": "^(.*)$",
                                "target": "$1",
                                "service": "html5-apps-repo-rt",
                                "authenticationType": "xsuaa"
                }]
}
```

3.2.2.4.3 Extend the View

Context

The view contains an input field, a button, and a text field. By pressing the button, a user starts a workflow instance. The value of the input field will be used in the workflow context. The response of the workflow start request will be printed out in the text field.

Procedure

In the view XML file created in *webapp/view* folder of your application, substitute the existing page element with the following code:

Sample Code

```
<Page title="Workflow Start UI">
    <content>
        <VBox width="100%" direction="Column" id="__vbox0">
            <items>
                <Input width="100%" id="textInput" value="{/text}"/>
                <Button text="Start Workflow" width="100px" id="__button0"
press="startWorkflow"/>
                <Text text="{/result}" maxLines="0" id="__text0"/>
            </items>
        </VBox>
    </content>
</Page>
```

3.2.2.4.4 Extend the Controller

Procedure

In the `controller JS` file created in *webapp/controller* folder of your application, include the following functions as the fields of the second parameter of the `Controller.extend` function call:

- Implement Data Model Instantiation [page 130]
- Bind an Action to the Button Push Event [page 132]
- Implement Workflow Instance Start [page 131]
- Implement XSRF Token Fetch [page 130]

Results

As a result, the extension parameter of the controller looks as follows:

ℹ Note

- Configure the following code by providing the **<app id>**. This is the value of the **id** in the `sap.app` section of the `manifest.json` file. To find this section, start the *Code Editor* in SAP Web IDE Full-Stack or SAP Business Application Studio and open *MTA project* > *HTML5 module* > *webapp* > *manifest.json*.
- Remove each period from the value of **id**. For example, the <app id> value for `"id":"demo.sap.com.simpleTaskUI"` is **demosapcomsimpleTaskUI**.

Sample Code

```
{
    onInit: function() {
        this.getView().setModel(new sap.ui.model.json.JSONModel({
            text: "",
            result: ""
        }));
    },

    startWorkflow: function() {
        var token = this._fetchToken();
        this._startInstance(token);
    },

    _startInstance: function(token) {
        var model = this.getView().getModel();
        var text = model.getProperty("/text");
        var contextJson = JSON.parse(text);
        $.ajax({
            url: "/<app id>/bpmworkflowruntime/v1/workflow-instances",
            method: "POST",
            async: false,
            contentType: "application/json",
            headers: {
                "X-CSRF-Token": token
            },
            data: JSON.stringify({
                definitionId: "<your workflow ID>",
                context: contextJson
            }),
            success: function(result, xhr, data) {
                model.setProperty("/result", JSON.stringify(result, null, 4));
            }
        });
    },

    _fetchToken: function() {
        var token;
        $.ajax({
            url: "/<app id>/bpmworkflowruntime/v1/xsrf-token",
            method: "GET",
            async: false,
            headers: {
                "X-CSRF-Token": "Fetch"
            },
            success: function(result, xhr, data) {
                token = data.getResponseHeader("X-CSRF-Token");
            }
```

```
        });
        return token;
    }
}
```

3.2.2.4.4.1 Implement Data Model Instantiation

During initialization a data model should be assigned to the view. In this example, the model is represented by an object with two fields: *text* and *result*. The *text* field refers to the input of the user, which will be used in the workflow instance context while starting. The *result* field refers to the string representation of the response to the workflow start request:

Sample Code

```
onInit: function() {
    this.getView().setModel(new sap.ui.model.json.JSONModel({
        text: "",
        result: ""
    }));
}
```

3.2.2.4.4.2 Implement XSRF Token Fetch

To call the workflow start REST API, the request needs an XSRF token. The following function can supply the token:

Note

- Configure the following code by providing the **<app id>**. This is the value of the **id** in the `sap.app` section of the `manifest.json` file. To find this section, start the *Code Editor* in SAP Web IDE Full-Stack or SAP Business Application Studio and open *MTA project* > *HTML5 module* > *webapp* > *manifest.json*.
- Remove each period from the value of **id**. For example, the <app id> value for `"id":"demo.sap.com.simpleTaskUI"` is **demosapcomsimpleTaskUI**.

Sample Code

```
_fetchToken: function() {
    var token;
    $.ajax({
        url: "/<app id>/bpmworkflowruntime/v1/xsrf-token",
        method: "GET",
        async: false,
        headers: {
            "X-CSRF-Token": "Fetch"
        },
        success: function(result, xhr, data) {
```

```
            token = data.getResponseHeader("X-CSRF-Token");
        }
    });
    return token;
}
```

3.2.2.4.4.3 Implement Workflow Instance Start

The workflow is started using the corresponding HTTP call to the workflow service REST API.

For more information, see Using Workflow APIs [page 151]. In this example, the input of the user is used in the context of the workflow instance: Namely, in its text field. In addition, the response of the call is assigned to the corresponding property in the data model:

> **i Note**
>
> - Configure the following code by providing the **<app id>**. This is the value of the **id** in the `sap.app` section of the `manifest.json` file. To find this section, start the *Code Editor* in SAP Web IDE Full-Stack or SAP Business Application Studio and open *MTA project* > *HTML5 module* > *webapp* > *manifest.json*.
> - Remove each period from the value of **id**. For example, the <app id> value for `"id":"demo.sap.com.simpleTaskUI"` is **demosapcomsimpleTaskUI**.

Sample Code

```
_startInstance: function(token) {
    var model = this.getView().getModel();
    var inputValue = model.getProperty("/text");
    $.ajax({
        url: "/<app id>/bpmworkflowruntime/v1/workflow-instances",
        method: "POST",
        async: false,
        contentType: "application/json",
        headers: {
            "X-CSRF-Token": token
        },
        data: JSON.stringify({
            definitionId: "<your workflow ID>",
            context: {
                text: inputValue
            }
        }),
        success: function(result, xhr, data) {
            model.setProperty("/result", JSON.stringify(result, null, 4));
        }
    });
}
```

> **i Note**
>
> Substitute the `<your workflow ID>` part of the URL with the ID of the deployed workflow definition of interest.

Feel free to change the name of the text field of the workflow context to fit to the corresponding workflow definition.

3.2.2.4.4.4 Bind an Action to the Button Push Event

The logic described above is triggered when a user presses the button:

Sample Code

```
startWorkflow: function() {
    var token = this._fetchToken();
    this._startInstance(token);
}
```

3.2.3 Creating a Workflow Form

End users can interact with a workflow through user interfaces.

- Start Forms
  Initiate a workflow based on form input. To provide a start form to your end users, integrate it into your SAP Fiori launchpad. See Configure a Start-Form-Based Workflow Start App [page 188].
- Task Forms
  Enable end users to participate in a workflow instance using tasks in their inboxes.

> **Note**
>
> You can't use a start form in a user task or a task form to initiate a workflow.

A form includes a header section and a details section. The information that is displayed in the header depends on the form type:

- Start Forms
  The header information comes from the attributes that are configured in the SAP Fiori launchpad tile. See Configure a Start-Form-Based Workflow Start App [page 188]. Task-related attributes such as *Created On* and *Created By* aren't supported.
- Task Forms
  The header information comes from the runtime attributes of the user task's that are defined in the workflow editor, for example, *Created On* and *Created By*. In addition, some information comes from the workflow editor, for example, *Name*, *Subject*, and *Description*.

The *Form Details* section displays the UI definition that you set up in the form editor. You can model fields and also define a layout by grouping the fields into sections and subsections.

The footer bar renders the start action of a start form or the decisions of a task form.

Related Information

Create a Start Form and its Custom Tile for Your Workflow

3.2.3.1 Create Your Form

You can create forms using the form editor.

Create Your Form with SAP Business Application Studio

Procedure

1. In your dev space, choose *F1*, search for **`Form`** and select *Workflow: Create New Form*.
2. Enter the required details and confirm.

Field	Sample Value
Name	**`ApprovalForm`**
ID	**`approval-form`**
Revision	**`2.0`** or **`Draft`**
Type	**`Start Form`** or **`Task Form`**

Results

A corresponding file with the name `<yourformname>.form` is created, and the form editor opens an empty form.

To reopen a form file, either double-click it or choose *Form Editor* from the context menu.

You can rename the file at any time.

> **i** Note
>
> - If you've already referenced a task form file within a user task in a workflow, make sure to adapt the reference in the workflow editor accordingly.
> - The form ID must be unique inside your account. Don't change the form ID unless you're sure that you want to give the form a new identity.

Create Your Form with SAP Web IDE

Procedure

1. Right-click the workflow module or any folder within the module, for example, a dedicated forms folder, and choose *New* > *Form*.

   > **i Note**
   >
   > You can store your forms in any existing folder, or you might want to create a folder that's used only for storing the forms.

2. Enter a name, ID, and a revision for your form, for example:

Field	Sample Value
Name	`ApprovalForm`
ID	`approval-form`
Revision	`2.0` or `Draft`
Type	`Start Form` or `Task Form`

3. Choose *Create*.

Results

A corresponding file with the name `<yourformname>.form` is created, and the form editor opens an empty form. To reopen a form file, either double-click it or choose *Form Editor* from the context menu. You can rename the file at any time.

> **i Note**
>
> - If you've already referenced a task form file within a user task in a workflow, make sure to adapt the reference in the workflow editor accordingly.
> - The form ID must be unique inside your account. Don't change the form ID unless you're sure that you want to give the form a new identity.

3.2.3.1.1 Define Form

You can build your form by using fields or collections. You can arrange it with sections and subsections.

Related Information

Add Fields [page 135]
Add Collections [page 140]

3.2.3.1.1.1 Add Fields

Build forms using fields that you can arrange using sections and subsections.

Procedure

1. To open a file, double-click it.
2. To add a field to your form, choose *Add Field* at the top of the field table.

   Where the field appears depends on whether you've selected an existing field, any sections, subsection, or collection.

   If you selected an existing field before choosing *Add Field*, then the new field is inserted right below the selected field. If you don't select an existing field, the new one that's added depends on which element you've selected. If a section is selected, the new field is added at the end of the section. If a subsection is selected, the new field is added to the end of the subsection. If a collection is selected, the new field is added to the end of the collection. For more information, see Adapt the Form Layout [page 143] and Add Collections [page 140].
3. On the *Properties* view, name the field by entering text for the field label.
4. Enter the ID of the field or use the automatically generated one.

   > **i Note**
   >
   > IDs must start with a letter and can contain only alphanumeric characters and underscores. IDs must be unique within a section, subsection, or collection. If a form doesn't contain sections, subsections or collections, IDs must be unique within the entire form.

5. Bind your field to a property element of the context model.

   For a start form, the context path refers to the workflow context, whereas for a task form it refers to the task context.

   Start Form

You can bind fields in start forms almost in the same way as in task forms. However, the workflow context is part of a running workflow instance. Start forms are used to initiate a workflow. At this point in time when a start form renders, no workflow context exists and fields of your start form don't show any initial value. A workflow start creates the bound properties in the empty workflow context. For more information, see Automatic Model Initialization [page 146].

Task Form

When you bind a field to a property in the task context, the respective value is shown during form rendering. Furthermore, if the field is set to editable (see Set the mode of your field [page 138]), changes to that value by the user are written back to the task context during task completion. If a bound property doesn't exist in the task context, it's created during task completion. For more information and limitations, see Automatic Model Initialization [page 146].

The syntax follows the JUEL style described in Expressions [page 78]. Fields on form level, within sections or subsections are bound to an absolute context path within the context model (keyword: 'context'). For fields that are part of collections, you usually specify a path relative to the collection's context path (keyword: 'item'). For more information, see Add Collections [page 140].

Context Path Suggestions

The form editor helps you enter complex context path bindings. This requires a syntactically valid JSON, for example, your workflow sample context. It must exist on the same level as your form definition and must have the same name, for example, `my.form` and `my.json`.

As a result, the editor provides a list of suggestions based on the already entered characters and the underlying JSON. In addition, sample values are shown.

> **i Note**
>
> You can access the context model only using dot notation. Conditions and literals aren't supported. Make sure that you use a valid path to a property in the context.

> **Example**
>
> Let's take a sample task form and assume that your task context is the following:

> **Sample Code**

```
{
  "report": {
    "name": "Travel for TechEd Las Vegas",
    "id": "A2E6D6A5ABD4C37",
    "owner": "Steve Consultant",
    "totalClaimedAmount": 870.30,
    "currencyCode": "EUR",
    "includesVAT": true,
    "numItems": 5,
    "invoices": [{
      "date": "2017-10-09",
      "time": "13:30:00",
      "orderDateTime": "2017-10-01T09:15:43.000Z",
      "amount": 420.0
    },
    {
      "date": "2017-10-15",
```

```
        "time": "09:15:00",
        "orderDateTime": "2017-10-10T14:33:21.000Z"
        "amount": 510.0
      }]
    }
}
```

Sample Code

You want to define a field within a section that displays the timestamp of the purchase order in the invoice. You can use the following data for this field:

Property	Sample Value	Comment
Label	Date and Time of Purchase Order	-
Type	DateTime	-
Context Path	`${context.report.invoices[0].orderDateTime}`	The context path points to the property orderDateTime within the first item of the invoices array.

For an example for collection fields, see Add Collections [page 140].

6. Set the type for your field.

   Field types determine how the field is represented in your task UI. The task UI validates the user input against the value range of the specified type.

   Currently the following field types are supported:

Type	Value Range	UI Representation
String	Any printable character	Labeled input field
Boolean	false or true	Checkbox
Integer	-9007199254740991 to 9007199254740991	Labeled input field
Float	-3.4028235e+38 to 3.4028235e+38	Labeled input field
Date	Any date of Gregorian calendar from year 1 to year 9999	Labeled input field with date picker
DateTime	Any point in time within a date	Labeled input field with date and time selector
Time	Any time of a day, from 00:00:00 to 23:59:59 (or 12:00:00AM-11:59:59PM depending on the locale)	Labeled input field with time selector

> **i Note**
>
> If the defined field type doesn't match the type of the actual value within the task context, the task form isn't rendered. A detailed error message is issued in the browser console.

7. Task form only. Set the mode of your field.

   Currently the following modes are supported for fields in a task form:

Mode	UI Representation
Editable (Default)	The end user can modify the value on the UI.
Display-Only	The end user isn't allowed to modify the value on the UI.

> **i Note**
>
> At this point in time when a start form renders, no workflow context exists and no value of a bound property can be shown. As a consequence, fields within a start form are always in *Editable* mode.
>
> If the complete form is set to read-only mode, it can't be changed (see Create Your Form [page 133]).
>
> The mode affects only the rendered form and not the workflow runtime itself. Read-only attribute values can still be modified, for example, using the REST API or script tasks.
>
> For fields that are part of a collection, you can set the display-only mode solely when adding and deleting items to the collection itself is disabled. See Add Collections [page 140].

8. Set the constraints of your field.

   Currently, only the following constraint is supported for editable fields:

Constraint	UI Representation
Required	The end user must enter a value; otherwise, they can't use decisions to complete the task (see Add or Delete Decisions [page 144]).

> **i Note**
>
> Constraints affect only the rendered form and not the workflow runtime itself. Required attribute values can still be set to an empty string using the REST API or script tasks.

9. (Optional) On the *Properties* under *UI Configuration*, change the standard UI control derived from the field type.

Field Type	Supported Controls (for editable fields)
String	Input, text area, dropdown, radio buttons
Boolean	Checkbox (can't be changed)

Field Type	Supported Controls (for editable fields)
Date	Input, dropdown, radio buttons
Time	Input, dropdown, radio buttons
DateTime	Input, dropdown, radio buttons
Integer	Input, dropdown, radio buttons
Float	Input, dropdown, radio buttons

Depending on the selected control, additional configuration options apply.

a. Define a placeholder for your *Input* or *Text Area*.

   That way, an empty control displays the placeholder to give users a hint when they enter data.

a. Set the height for your field. This is only available for fields that are of type "String" and aren't part of a collection.

   Currently the following field heights are supported:

Height	UI Representation
Single Line (Default)	A single-line input field
Small	A text area approximately twice the height of a single-line field with scrolling capabilities
Medium	A text area approximately twice the height of a small field with scrolling capabilities
Large	A text area approximately twice the height of a medium field with scrolling capabilities

b. Use dropdown lists or radio buttons.

   Selectable Values:

   Define an enumeration of allowed values for this type.

   Each entry consists of two parts:

   - The value that is being populated to the workflow context, if selected.
   - A human-readable display value that is used for presentation inside the control.

   For both the type-specific validations apply (see Form Validation [page 149]).

   - To add an entry, choose *Add* from the toolbar menu. You can add at most 100 entries to the list.
   - To remove a selected entry, choose *Delete* from the toolbar menu. You need at least 1 entry in the list.
   - To influence the order of a selected entry, choose *Move Up* or *Move Down* from the toolbar menu.

You can't mark fields that use dropdowns or radio buttons as optional. These fields are automatically set to "required" in the editor and you can't change this setting. To set the constraint yourself, you need to switch back to the default control setting.

> **i** Note
>
> If you use the form inside a task, the value inside the specified context path must match with one of the elements. Otherwise, the form isn't rendered.

3.2.3.1.1.2 Add Collections

Build task forms using collections that you can arrange using sections and subsections.

Context

Similar to sections and subsections, collections can contain fields. Collections are rendered in the form as tables, and the fields within a collection represent the table columns. As opposed to sections or subsection, collections themselves are bound to a context path, namely an array within the collection, which specifies the rows of the table.

Procedure

1. To open a file, double-click it.
2. To add a collection to your form, choose *Add Collection* at the top of the field table.

   Where the collection appears depends on whether you've selected an existing collection, any sections, or subsections. If you selected an existing collection before choosing *Add Collection*, then the new collection is inserted right below the selected collection. If you don't select an existing collection, the position of the newly added collection depends on which element was selected. If a section is selected, the new collection is added at the end of the section. If a subsection is selected, the new collection is added to the end of the subsection. For more information, see Adapt the Form Layout [page 143].
3. (Optional) In the *Properties* view, name the collection by entering text for the collection title, which would be rendered as the table title in the form.
4. Enter the *ID* of the collection or use the automatically generated one.

   > **i** Note
   >
   > IDs must start with a letter and can contain only alphanumeric characters and underscores. IDs must be unique within a section or subsection. If a form doesn't contain sections or subsections, IDs must be unique within the entire form.
5. Bind your collection to a property element. Collections on form level within sections or subsections are bound to an absolute context path within the task model (keyword: 'context'). The context path must point

to an array. This binding specifies the rows of your table. The syntax follows the JUEL style described in Expressions [page 78].

Example

Let's assume that your process context is the following:

```
{
  "report": {
    "name": "Travel for TechEd Las Vegas",
    "id": "A2E6D6A5ABD4C37",
    "owner": "Steve Consultant",
    "totalClaimedAmount": 870.30,
    "currencyCode": "EUR",
    "includesVAT": true,
    "numItems": 5,
    "invoices": [{
      "date": "2017-10-09",
      "time": "13:30:00",
      "orderDateTime": "2017-10-01T09:15:43.000Z",
      "amount": 420.0
    },
    {
      "date": "2017-10-15",
      "time": "09:15:00",
      "orderDateTime": "2017-10-10T14:33:21.000Z"
      "amount": 510.0
    }]
  }
}
```

You want to define a collection that displays a list of invoices. You can use the following data for this collection:

Property	Sample Value
Title	List of Invoices
Context Path	${context.report.invoices}

Note

You can access the context model only using dot notation. Conditions and literals aren't supported. Make sure that you use a valid path to a property in the context.

6. A collection is rendered as a table. While the bound context property of the collection specifies the rows of the table, you have to add fields to specify the columns. To do so, select the collection and press *Add Field*. For more information, see Add Fields [page 135]. For fields in collections, you can specify the context path relative to the containing collection. To do so, use the relative syntax `${item.path.to.relative.property}` instead of the absolute syntax `${context.path.to.property}`. However, you can still bind to absolute context paths.

Example

Using the same task context and collection as before, you want to specify that for each invoice, the timestamp of its purchase order and its amount should be shown. Also, each row should display the global currency code of the report.

You can use the following data for the collection fields:

Label	Context Path	Type
Date and Time of Purchase Order	`${item.orderDateTime}`	DateTime
Amount	`${item.amount}`	Float
Currency	`${context.report.currencyCode}`	String

3.2.3.1.2 Set the Form Mode

Once you've created a form, you can choose whether end users are allowed to change the values in the form's fields.

Context

This only applies to task forms. At the point in time when a start form renders, no workflow context exists and no value of a bound property can be shown. As a consequence *Form Mode* isn't supported for start forms and all fields within a start form are always in *Editable* mode.

Procedure

1. In the fields table, make sure that no field is selected.

   > **i Note**
   >
   > To deselect a selected field, click it again.

2. Set the mode of your form.

   Currently, the following modes are supported:

Mode	UI Representation
Editable (Default)	For each field, you can modify the value on the UI.
Display-Only	For each field, you can no longer modify the value on the UI. The fields are in display mode.

A form-wide display-only mode overwrites individual field "display-only" modes. You can no longer change the mode for individual fields. However, your previously modeled modes as well as constraints on the individual fields are preserved in case you switch to form wide *Edititable* mode again.

> **i Note**
>
> The mode affects only the rendered form and not the workflow runtime itself. You can still modify read-only attribute values at the API level or in script tasks.

3.2.3.1.3 Adapt the Form Layout

Define the layout of your forms, for example, whether to group fields.

Group Fields or Collections into Sections and Subsections

To group your fields or collections, choose *Add Section*. If your form does not have any sections, this action moves all fields or collections into a new section. Otherwise, a new section is added.

If you need a more granular grouping, you can choose *Add Subsection* while a section is selected. If the selected section already contains fields or collections, they are moved into the new subsection.

> **i Note**
>
> Don't add more than 100 sections to your form, or more than 100 subsections to a single section. If you need more than 100 subsections, divide them up across several sections.
>
> If you need more than 100 sections, split your UI into multiple forms that are connected using multiple user tasks.
>
> Only 100 fields or collections per section or subsection are supported. It is not possible to have both collections and fields next to each other in the same section or subsection.

To add new fields to a section or subsection, select it and choose *Add Field*. To add new collections to a section or subsection, select it and choose *Add Collection*.

For each section or subsection, you can specify text for a section title.

Move Form Elements

Use the following options to change the location of fields, sections, and subsections:

- The context menu
- The actions in the table toolbar (copy, cut, paste)
- Drag and drop

If you paste a subsection before or after a section, the subsection is converted into a section. If you paste a section before or after a subsection or into a section, then it's converted into a subsection. The latter one applies only to sections that contain fields but not to sections that contain subsections. By using copy and paste, it is also possible to move elements from one form to another.

Pasting collections or fields that are part of a collection into a start form is not possible.

3.2.3.1.4 Create Decisions of a Task Form

Users can complete tasks using decision buttons.

You can model the following types of decisions for your task form:

- A positive decision
  Example: Approve
- A negative decision
  Example: Reject
- A neutral decision

Each decision type has its own visual appearance that matches its semantics.

The workflow context stores the decision the user has selected. For more information about how to access the decision, see Access the Decisions [page 145].

3.2.3.1.4.1 Add or Delete Decisions

You must define the decisions that users can choose from to complete a task.

Procedure

1. Switch to the *DECISIONS* tab.
2. Choose *Add*.
   a. Specify the display text for the decision button.
   b. (Optional) Edit the generated decision ID.

      > **i Note**
      >
      > IDs must start with a letter and can contain only alphanumeric characters and underscores. Decisions in your form must have a unique ID.

   c. Specify the decision type.

Move Decisions

Procedure

You can duplicate or change the order of decisions using any of the following options:

- The context menu
- The actions in the table toolbar (copy, cut, paste)
- Drag and drop

> **i Note**
>
> My Inbox sorts the decisions first by type (Positive, Negative, Neutral), then by the order in which they're listed in the DECISIONS table.

3.2.3.1.4.2 Access the Decisions

To complete a task, an end user selects a decision.

Context

For user tasks that use forms, the decision is stored within the user task's properties. See Expressions [page 78]. For each completed task in a flow, the `decision_id` of the most recently selected decision is stored in the *decision* property.

> **Example**
>
> An end user chooses *Accept* in a user task with the ID "usertask1". You can access the corresponding decision ID, for example, `accept`, using a JUEL expression, as follows:

> **Sample Code**
>
> ```
> ${usertasks.usertask1.last.decision}
> ```

Procedure

Use the decision in the context of an exclusive gateway, see Configure a Sequence Flow [page 70]

> **Example**
>
> ```
> "${usertasks.usertask1.last.decision=="accept"}"
> ```

3.2.3.1.5 Automatic Model Initialization

There are a few runtime behaviors that you must consider when creating forms, for example, make sure that you avoid binding collisions.

Model Initialization

To ensure that a form renders correctly in the UI, you must build the corresponding data model so that it can be rendered. The following items are verified automatically:

- Binding collisions
  When a binding collision occurs, the UI doesn't render, and shows an error message. In addition, the workflow forms runtime posts an aggregated issue report to the browser console.
- Referenced objects in binding path are missing.
  You can bind fields to context properties that don't exist in the task context when the form is opened. The workflow forms runtime creates the missing context properties.

> **i Note**
>
> The missing properties are created with a default value, depending on their type:

Property Type	Default Value
Numerical values, time, date and datetime	null
String	empty string ("")
Boolean	false

> **i Note**
>
> For binding paths that include array elements, missing context properties are created only if each array element in the path exists in the context.
>
> As for start forms no workflow context exists during rendering, binding to array elements isn't supported.

For more information, see Bind your field to an attribute of the task context model [page 135]. This is what it looks like:

Workflow (Task) Context	Binding	UI Model (Initial)	UI Model with Data
{}	${context.myNode1.my-Node2.myProperty}	`{ myNode1: { myNode2:{} } }`	`{ myNode1: { myNode2: { myProperty: "anyValue" } } }`

Task forms only. Missing context properties are also created for an existing element in a array.

Workflow (Task) Context	Binding	Input	UI Model with Data
{a : [{}]}	${context.a[0].b.c}	`anyValue`	`{ a: [{ b: { c: "anyValue" } }] }`

For example, the following scenario is not supported, because the specified array element (0) doesn't exist:

Workflow (Task) Context	Binding
{a : []}	${context.a[0].b.c}

3.2.3.1.6 Versioning Forms

Versioning is a key activity that should be considered by all developers who build production-grade software.

This holds specifically true for forms used by potentially long-running workflows. Without versioning, changes you develop for forms in future workflow instances unexpectedly also affect already running instances. These unintended changes often have a negative impact.

Comparison to Versioning with Git or Similar Tools

Don't confuse versioning of forms with other versioning methods that use version control systems (VCS). Versioning forms in a Git repository handles design-time versioning of artifacts and is orthogonal to the runtime-related versioning discussed here. See below for recommendations on how to combine the two.

Technical Versioning of Forms

By default, forms are already versioned in a technical way: Each time a form is deployed to the runtime, a new (technical) version is created for it. Previous versions are preserved for historical and auditing reasons; however, end users cannot access them at runtime. This way, developers and administrators have transparency over who deployed which form and when.

Compatible Changes Compared to Incompatible Changes

In the technical versioning outlined above, any change to a form represents a new (unqualified) version. There's no way for developers to distinguish between compatible and incompatible changes.

If a change or release fixes a usability or functional issue, it's typically considered compatible. Incompatible changes fundamentally alter a form and are usually driven by a business requirement. An incompatible change can, for example, apply to mandatory form fields that you add to a form. Consequently, a workflow needs to store additional data in its context that is expected by the changed form. To address this, developers typically need to change the workflow definition accordingly. Incompatible changes are assumed to take effect for new workflow instances while already running workflow instances continue to operate on the previous version. Already running workflow instances wouldn't have the necessary context data.

Forms Revision Concept

To allow the differentiation between compatible and incompatible changes, each form has a revision property that is stored along with any other properties, for example, the form name and form ID.

You can set the revision property when you create a new form or edit its metadata. For more information, see Create Your Form [page 133].

When you refer to a form in a workflow's user task, you are asked to specify the revision of the form to use. For more information, see Configure a User Task UI Using Workflow Forms [page 38].

As stated above, changing the revision of a form and deploying it to the form runtime implies a major release of the form. By contrast, deploying a form without a change of its revision implies a minor release. This lets you choose between changes that affect existing workflow instances and changes that affect only future workflow instances, provided that you change the revision of the respective workflows' user tasks accordingly.

Versioning Best Practices

The following is a list of recommendations of when to leave a form revision unchanged, and when to alter the revision.

Compatible Changes (Revision Unchanged)	Incompatible Changes (Revision Altered)
Change the label or placeholder for a form field	Change a form field type
Change the layout settings for a form, for example, sections or subsections	Change the ID or value for a form field (*)
Change the text of a form field	Change the decision ID for a form field
Remove a read-only form field	Add or remove a form field (*)
	Add a read-only form field (*)

(*) Although there may be conditions where compatibility can be ensured, these types of changes are usually incompatible. This is decided on a case-by-case basis.

> → Tip
>
> When you're changing a form revision, we recommend that you tag or otherwise flag the corresponding commits in Git. This helps when you need to patch an older revision at a later time.

Related Information

Add Fields [page 135]

3.2.3.1.7 Form Validation

The form editor automatically validates your form while you model it.

If there are missing mandatory entries or invalid inputs for any of your form's elements, for example, for fields, sections, subsections, or decisions, the form editor notifies you about these inputs. This already happens when you add an element to your form.

You receive information about errors in your modeled form as follows:

- Form elements with invalid inputs are highlighted with a red mark.
- If the element itself has valid inputs but contains at least one other element with invalid inputs, an orange mark is displayed.
- The actual validation error for each input of the form element is shown in the respective properties view. Invalid input fields are highlighted in red and have an error message attached.

> ℹ Note
>
> If the form editor detects any error in a form, it also adds a reference to the *Problems* view. The *Problems* view is dynamically updated when you change files within the scope of the analysis.

3.2.3.1.8 Work with Attachments

SAP Workflow service a lightweight integration with SAP Document Management service. This means that binary files are completely handled by SAP Document Management service (storage, access management, and more) while the workflow service maintains the relation between a workflow instance, its files, and additional metadata. You can choose whether in a form, end users are allowed to see attachments related to a workflow instance. The feature provides read-only access to attachments that were added to the workflow using the API.

For more information, see Work with Attachments on a Workflow and Task Instance [page 160].

Currently, attachments are only available for task forms. As the feature just supports "read-only", no attachments are rendered in a start form.

Please take the following into consideration:

- Missing Repository Connectivity
  The workflow service only manages the relation between workflow and attachment. Additional details are retrieved from the configured (document) repository. If the configuration is (temporarily) unavailable, the form still displays the attachments, but attributes like name, creator, or size might not be shown. Furthermore, attachments cannot be downloaded.
- Missing Attachments In Form
  Currently forms only render attachments that are added to the "default" group. Attachments that are added to different groups are not shown.
- Modifying Attachments Using the UI
  This feature provides read-only access. Therefore, attachments must be added, removed, or altered using the API.

3.2.3.1.8.1 Enable or Disable Attachments

You can choose whether end users are allowed to see attachments related to a workflow instance inside a form.

Prerequisites

- The calling user has the correct permissions to invoke the APIs.
- SAP Document Management service for workflow service attachments is configured, see Configure Document Management for Workflow Service Attachments [page 180].

Procedure

1. In the fields table of your task form, make sure that no field is selected.

   > **i Note**
   >
   > To deselect a selected field, click it again.

2. Enable or disable the attachments feature using the checkbox.

3.3 Using Workflow APIs

The REST-based API allows a tight integration of tasks on SAP Business Technology Platform (SAP BTP) with SAP Workflow service.

SAP Workflow service exposes two kinds of API to address different use cases. The OData-based APIs expose user-task related data implementing a subset of the Task Consumption Model (TCM), see SAP Note 2304317. Their primary use case is to build a personal inbox. The REST-based APIs allow you to list and manage workflow instances, definitions, and user tasks across recipients. Depending on your role, you can do the following:

- Send messages to workflows.
- List user task instances and inspect details of a user task instance and its context.
- List workflow definitions and inspect details of a workflow definition.
- List workflow instances and inspect details of a workflow instance, its context, and its execution log.
- Execute various lifecycle and administrative operations on the resources involved in the workflow service.

For information about who can execute these actions, see Authorization Configuration [page 229].

Access the API Documentation

See the SAP Workflow service API hub documentation.

The API documents are also uploaded individually:

- Workflow API for Cloud Foundry
- Inbox API for Cloud Foundry

Authentication and Authorization

Clients must authenticate to use the workflow service APIs. The following authentication types are supported:

- OAuth2 (authorization code, and SAML 2.0 Bearer Assertion Flow for OAuth 2.0)
  Certain authentication mechanisms are transparently managed by the application router for SAP BTP, Cloud Foundry environment applications that are bound to the workflow service.
  For example, the application router provides user-centric authentication mechanisms. For this purpose, it manages the current user's authorization tokens for the back-end services in the user session. When there is no user session, the user is redirected to the UAA's logon form.
- OAuth2 (Client Credential Grant for OAuth 2.0)
  For technical authentication, OAuth2 client credentials grant is supported.
  For example, an application might require a technical authentication without having a user authenticated through an identity provider. You can use the OAuth2 client credentials grant to authorize REST calls for

your tenant. For more information about how to grant authorizations to OAuth clients, see Technical Authentication [page 232].

> **i Note**
>
> For all OAuth2-based workflow APIs, you do not need to specify a CSRF token. This is because the OAuth2 flows already have CSRF protection when used without intermediaries. However, when workflow API requests are routed through an application router, you must apply CSRF token mechanisms. Otherwise, the CSRF protection is lost. Therefore, do not turn off the `csrfProtection` setting of routes in the application router that use `xsuaa` as `authenticationType`.

Libraries and Integration

The SAP Cloud SDK for Java provides an up-to-date, type-safe client library to consume the REST API of the workflow service. The SDK also simplifies the handling of destinations and authentification. In addition, it provides features for native support of SAP Business Technology Platform (SAP BTP). For more information, see SAP Cloud SDK for Java and type-safe client library .

Rate Limits

To ensure optimal operation of the service, REST API execution is subject to resource limits, for example, regarding the number of requests per second. If the limits are exceeded, API calls return `HTTP status 429` (“Too many requests”). The client should then reduce the number of calls.

3.3.1 Determine Service Configuration Parameters

In the SAP BTP, Cloud Foundry environment, you often require basic configuration parameters of the workflow service to access workflow APIs.

Prerequisites

You have the *Space Developer* permission in your subaccount.

Procedure

1. Navigate to the space in which you've created a service instance for which you want to view the service key. For more information, see Navigate to Orgs and Spaces.

2. In the navigation area, choose *Services* > *Service Instances*.
3. Click the link for your service.
4. In the navigation area, choose *Instances*, then select the service instance for which you want to view the key.
5. In the navigation area, choose *Service Keys*.
6. To view the configuration parameters, choose one of the following options.
   - If you work with a service key, you receive only the configuration parameters that are relevant for the workflow service.
     For more information, see Create Service Keys Using the Cockpit in the SAP Business Technology Platform (SAP BTP) documentation.
     See the table for details on the configuration parameters.
   - If you work with a service binding, you find all the configuration parameters that are relevant for the workflow service under the "workflow" top-level property.
     For more information, see Binding Service Instances to Applications in the SAP Business Technology Platform (SAP BTP) documentation.
     In a running application, the same information is usually available from the VCAP_SERVICES environment variable. See the table for details on the configuration parameters.

Parameter Name	Description	Example
`uaa.clientid`	Client ID for OAuth2	Not available
`uaa.clientsecret`	Client secret for OAuth2	Not available
`uaa.url`	Authentication base URL for OAuth2 or SAML	**`https://<subdomain>.authentication.<region host>.hana.ondemand.com`**
`endpoints.workflow_rest_url` and `endpoints.workflow_odata_url`	URL of the workflow service	**`https://api.workflow.<region-host>.hana.ondemand.com/workflow-service/rest`** and **`https://api.workflow.<region-host>.hana.ondemand.com/workflow-service/odata`**

3.3.2 Access Workflow APIs Using OAuth 2.0 Authentication (Authorization Code Grant)

This procedure illustrates how to call workflow service APIs using OAuth 2.0 authentication using an example walk-through of the authorization code flow. It shows how several OAuth2 concepts are applied to workflow service and which configuration parameters are used.

Prerequisites

- You've created a service instance of the workflow instance.
- You have bound the service instance to your SAP BTP, Cloud Foundry environment application, or you've created a service key. See Create a Service Key Using the Command Line Interface [page 173].
- You've noted down the following parameters from the procedure Determine Service Configuration Parameters [page 152]: `clientid`, `clientsecret`, `url`, `workflow_rest_url`, or `workflow_odata_url`.
- Assign the necessary role collections of the workflow service API that you want to call to the user on whose behalf the call to the workflow service API is executed. Typically, this is the user who authenticates the call to the OAuth 2.0 authorization endpoint.
  For more information on role collections, see Assign Workflow Roles to Your Users [page 19].
  For more information about roles, see Authorization Configuration [page 229].

Context

Developers typically use this flow in web applications. However, other flows might be supported or more appropriate in your use case. See, for example, a blog about another flow.

> **i Note**
>
> You must apply URL encoding to all the parameters that contain special characters, for example, the `clientid` and `redirect_uri` parameters.

Procedure

1. Request an authorization code from the OAuth 2.0 authorization server.
   a. Send a GET request to the authorization endpoint and specify both the client ID and the response type as "code". Use the `url` and `clientid` service configuration parameters.

      Example: Combine the parameters with the `authorize` end-point of the authorization server, for example, as follows: `<url>/oauth/authorize?client_id=<clientid>&response_type=code`.

      You can configure the URI to which the call redirects with the `redirect_uri` parameter. If you don't provide a value, a default one is set.

b. Authenticate the call using the real user that should be propagated to the workflow service API (on behalf user).

   The response redirects to the URL that you specified as callback URL in the `redirect_uri` parameter. The value of the parameter code represents the authorization code.

c. Copy the code from the HTTP URL.

2. Request an access token from the OAuth 2.0 authorization server.

   a. Send a POST request to the token endpoint and specify the grant type as authorization code. Use the `url` service configuration parameter and the code from step 1c.

      Combine the parameters with the `token` end-point of the authorization server, for example, as follows: `<url>/oauth/token?grant_type=authorization_code&code=<code_step1c>`.

   b. Authenticate the call using basic authentication, where the user name corresponds to your OAuth client ID and the password to the client secret.Use the respective service configuration parameters `clientid` and `clientsecret`.

   c. Copy the access token from the HTTP response body (`access_token` attribute of the JSON structure).

   > **i Note**
   >
   > If the access token expires before you get to execute step 3, use a refresh token.
   >
   > The HTTP response in step 2 includes a refresh token (`refresh_token` attribute). It's typically used when the lifetime of the returned access token has expired but the application still wants to execute an HTTP request (as in step 3) on behalf of the given user. You can use the refresh token to request a new access token for the user without again asking the user for consent. The new access token then replaces the old one with a new lifetime.
   >
   > To request a new access token for a given refresh token, send a POST request to the same token endpoint as in step 2 passing the refresh token. The call must be authenticated again with basic authentication, where the user name corresponds to your OAuth client ID and the password to the client secret.
   >
   > Combine the parameters with the `token` end-point of the authorization server, for example, as follows: **`<url>/oauth/token?grant_type=refresh_token&refresh_token=<refresh_token>`**
   >
   > However, it's important to understand that the refresh token has a lifetime as well. Lifetimes of access and refresh tokens can be configured separately. If the lifetime of the refresh token has expired, there's no means to request a new refresh token.

3. Perform the call to the workflow service API by sending the access token as the header. Use the end-points below the base URL from the service configuration parameter `workflow_rest_url or workflow_odata_url`.
   - Header name: `Authorization`
   - Header value: `Bearer <access token>`

3.3.3 Access Workflow APIs Using OAuth 2.0 Authentication (Client Credentials Grant)

You can call workflow service APIs using OAuth 2.0 authentication.

Prerequisites

- You've created a service instance of the workflow service with the necessary authorities to invoke the REST APIs as described in Technical Authentication [page 232].
- You have bound the service instance to your SAP BTP, Cloud Foundry environment application, or you've created a service key. See Create a Service Key Using the Command Line Interface [page 173].
- You've noted down the following parameters from the procedure Determine Service Configuration Parameters [page 152]: `clientid`, `clientsecret`, `url`, `workflow_rest_url`, or `workflow_odata_url`.

Context

This procedure shows an example walk-through of the client credentials grant. It shows how several OAuth2 concepts are applied to workflow service and which configuration parameters are used.

Developers typically use client credentials grant in technical scenarios without having a user authenticated through an identity provider. You can use the OAuth2 client credential grant to authorize REST calls for your tenant.

Procedure

1. Request an access token from the OAuth 2.0 server.
   a. Send a POST request to the token endpoint URL appended with the `?grant_type=client_credentials` parameter:

      ```
      curl \
      -X POST \
      <url>/oauth/token?grant_type=client_credentials \
      -u '<clientid>:<clientsecret>'
      ```

      Where <url> is the URL, <clientid> the client ID, and <clientsecret> the client secret you noted down as prerequisite from the service configuration.
   b. Note down the access token from the HTTP response body stored in the `access_token` attribute of the JSON structure.
2. Perform the call to the workflow service API by sending the access token as the header. Use the end-points below the base URL from the service configuration parameter `workflow_rest_url` or `workflow_odata_url`.

- Header name: `Authorization`
- Header value: `Bearer <access token>`

3.3.4 Determine the Service Host

In the SAP BTP, Cloud Foundry environment, the base URL of the workflow service is available from the `endpoints.workflow_rest_url and endpoints.workflow_odata_url` configuration parameter of the service key or of the service binding, depending on your application type.

For information on how to access this information, see Determine Service Configuration Parameters [page 152].

> **i Note**
>
> If you access the workflow APIs from a user interface of an application, you typically need to use a URL that enables Cross-Origin Resource Sharing (CORS) through reverse proxies.
>
> If, for example, you implement the Application Router, instead of directly referring to the URL defined from the above, you have to refer to application routes and destinations.

3.3.5 Modifying the Context of a Workflow Instance

You can modify a context of a workflow instance in RUNNING, ERRONEOUS, or SUSPENDED status.

> **i Note**
>
> - If the context of a workflow instance is in COMPLETED or CANCELED status, the system does not allow you to modify it.
> - We recommend suspending the workflow instance first and ensure that further entries are not written into the corresponding execution log. Then the context modification is considered safe from collisions with any ongoing workflow instance activities. After the necessary changes to the context are performed, you can resume the workflow instance execution. See the section about suspending or resuming workflow instance.
>   For more information, see /v1/workflow-instances/{workflowInstanceId}.

Override Context

Overriding a context of the workflow instance removes the contents of the context before performing the override operation. It is substituted with the payload of the operation.

> **Example**
>
> Context contents before overriding:

Sample Code

```
{
    variableOnlyInOldContext: 1,
    variableOverriden: "good bye!",
    variableNestedObject: {
            variableNested: true,
            variableNestedInOldContext: 1000
        }
}
Override operation payload:
{
    variableOverriden: "hello!",
    variableNestedObject: {
            variableNested: false,
            variableNestedNew: "new value"
        },
    variableNew: "I'm new"
}
```

Context contents after the override operation (equals the payload of the override operation):

Sample Code

```
{
    variableOverriden: "hello!",
    variableNestedObject: {
            variableNested: false,
            variableNestedNew: "new value"
        },
    variableNew: "I'm new"
}
```

Patch Context

Patching a context of the workflow instance merges the contents of the context before performing the override operation with the payload of the operation.

The following situations are possible in this case:

- A variable is present in the workflow instance context and in the operation payload. After the operation is performed, the value of this variable in the workflow instance context is equal to the corresponding value in operation payload.
- A variable is present in the workflow instance context, but not in the operation payload. After the operation is performed, the variable remains unchanged.
- A variable is not present in the workflow instance context before performing the operation, but it is present in the operation payload. After the operation is performed, the variable is added in the workflow instance context with the corresponding value.

> **Note**
>
> Merging happens at all levels of complex objects nesting.

Example

Context contents before patching:

Sample Code

```
{
    variableOnlyInOldContext: 1,
    variableOverriden: "good bye!",
    variableNestedObject: {
            variableNested: true,
            variableNestedInOldContext: 1000
        },
    variableNew: "I'm new"
}
```

Patch operation payload:

Sample Code

```
{
    variableOverriden: "hello!",
    variableNestedObject: {
            variableNested: false,
            variableNestedNew: "new value"
        },
    variableNew: "I'm new"
}
```

Context contents after the override operation:

Sample Code

```
{
    variableOnlyInOldContext: 1,
    variableOverriden: "hello!",
    variableNestedObject: {
            variableNested: false,
            variableNestedInOldContext: 1000,
            variableNestedNew: "new value"
        },
    variableNew: "I'm new"
}
```

Consider the naming conventions for context variables. For more information, see Conventions, Restrictions, and Limits [page 8].

3.3.6 Work with Attachments on a Workflow and Task Instance

SAP Workflow service provides APIs to manage relations between a workflow instance and attachments that are uploaded to SAP Document Management service.

The following steps provide basic interaction examples and point to the detailed API documentation allowing for integration of attachments in custom start or task UIs.

Prerequisites

- SAP Document Management service for workflow service attachments is configured, see Configure Document Management for Workflow Service Attachments [page 180].
- The calling user has the correct permissions to invoke the APIs.

1. Add Attachment Information on Workflow Start

When starting a workflow instance using an API, an additional property can be sent using the payload, including information about attachments relevant for this workflow instance.

```
POST /v1/workflow-instances
Sample request:
{
  "definitionId": "...",
  "context": {},
  "attachments": {
    "rootFolder": "folder-meant-for-uploads",
    "groups": {
      "default": {
        "folder": "folder-meant-for-uploads-for-this-group",
        "refs": [
          { "objectId": "id1" },
          { "objectId": "id2" }
        ]
      }
    }
  }
}
```

> **i Note**
>
> You must upload the attachments yourself, keep track of the object IDs of the CMIS-compliant repository, and add them in the payload accordingly.
>
> We recommend that you use the destination that uses CMIS 1.1 browser binding inside your custom application, see Configure Document Management for Workflow Service Attachments [page 180].
>
> For more information, see https://help.sap.com/doc/80205e0dc75945538b451284fdcc935b/Cloud/en-US/wfs-core-api-docu-cf.html#api-WorkflowInstances-v1WorkflowInstancesPost.

2. Read Attachment Information from Workflow Instance

To render a list of attachments, you can retrieve the information for a particular workflow instance using the API:

```
GET /v1/workflow-instances/<instance-id>/attachments
Sample response:
{
  "rootFolder": "folder-meant-for-uploads",
  "groups": {
    "default": {
      "folder": "folder-meant-for-uploads-for-this-group",
      "refs": [
        { "objectId": "id1" },
        { "objectId": "id2" }
    ]
   }
  }
}
```

> **i Note**
>
> Additional information of an attachment, such as name, creation/modification dates, owner, must be queried (for example, asynchronously) from SAP Document Management service. The same holds true for generating a link to download a file.
>
> For more information, see https://help.sap.com/doc/80205e0dc75945538b451284fdcc935b/Cloud/en-US/wfs-core-api-docu-cf.html#api-WorkflowInstances-v1WorkflowInstancesWorkflowInstanceIdAttachmentsGet.

3. Read Attachment Information from Task Instance

A more common use case might be reading this information while operating on a task instance. The API looks similar to:

```
GET /v1/task-instances/<instance-id>/attachments
Sample response:
{
  "rootFolder": "folder-meant-for-uploads",
  "groups": {
    "default": {
      "folder": "folder-meant-for-uploads-for-this-group",
      "refs": [
        { "objectId": "id1" },
        { "objectId": "id5" },
        { "objectId": "id6" }
     ]
   }
  }
}
```

> **i Note**
>
> Additional information of an attachment, such as the name, creation/modification dates, owner, must be queried (for example, asynchronously) from SAP Document Management service. The same holds true for generating a link to download a file.

For more information, see https://help.sap.com/doc/80205e0dc75945538b451284fdcc935b/Cloud/en-US/wfs-core-api-docu-cf.html#api-UserTaskInstances-v1TaskInstancesTaskInstanceIdAttachmentsGet.

4. Update Attachment Information on Task Completion

When operating on a user task instance, you can add or remove attachments. Upon task completion, you must update the attachments information. For example:

```
PATCH /v1/task-instances/<instance-id>
Sample request:
{
  "status": "COMPLETED",
  "decision": "..."
  "context": {},
  "attachments": {
    "rootFolder": "folder-meant-for-uploads",
    "groups": {
      "default": {
        "folder": "folder-meant-for-uploads-for-this-group",
        "refs": [
          { "objectId": "id1" }
        ]
      }
  }
}
```

The "merge" behavior with existing data is as follows:

- Groups that are part of the payload are overwritten completely.
- Groups that are not part of the payload but exist in the backend remain untouched.

This means, that "removing" a group on task completion is not possible, but making it empty is.

5. Override Attachment Information

For administrative use cases, there is an API that allows you to override the complete attachments information. For example:

Sample Code

```
PUT /v1/workflow-instances/<instance-id>/attachments
Sample request:
{
  "rootFolder": "folder-meant-for-uploads",
  "groups": {
    "default": {
      "folder": "folder-meant-for-uploads-for-this-group",
      "refs": [
        { "objectId": "id1" },
        { "objectId": "id5" },
        { "objectId": "id6" }

    ]
  }
```

```
}
```

> **Note**
>
> While using this API, you might change the data in such a way that subsequent workflow steps cannot react properly to it.
>
> For more information, see see https://help.sap.com/doc/80205e0dc75945538b451284fdcc935b/Cloud/en-US/wfs-core-api-docu-cf.html#api-WorkflowInstances-v1WorkflowInstancesWorkflowInstanceIdAttachmentsPut.

3.3.7 Updating Task Properties

With the task patch API, you can modify the properties of the tasks in status READY or RESERVED.

See task patch API.

To update a task, send an HTTP request with the `PATCH` method to the corresponding API endpoint. Use, for example, the following payload:

> **Sample Code**

```
{
                "subject": "<New subject>",
                "description": "<New description>",
                "dueDate": "<New due date>",
                "priority": "<New priority>",
                "processor": "<New processor>",
                "recipientUsers": "<New recipient users>",
                "recipientGroups": "<New recipient groups>"
}
```

Where `<New subject>`, `<New description>`, `<New due date>`, `<New priority>`, `<New processor>`, `<New recipient users>`, and `<New recipient groups>` refer to the values of the task subject, description, due date, priority, processor, recipient users, and recipient groups after the operation is performed.

Although this sample includes all fields, you only need to specify those fields that you really want to change.

For the `"priority"` field, the following values are supported:

- `"LOW"`
- `"MEDIUM"`
- `"HIGH"`
- `"VERY_HIGH"`

You can specify the due date using either of these formats: `yyyy-MM-dd'T'HH:mm:ss[.SSS]'Z'` or `yyyyMMddHHmmss[.SSS]`. The specified time stamp is UTC.

Supported date values are, for example:

- 2018-02-17T12:28:51Z

- 2018-02-17T12:28:51.854Z
- 20180217122851
- 20180217122851.854

> **Note**
>
> The workflow service does not explicitly check whether processors, recipient users, or groups assigned to user tasks actually exist in the system. For example:

Sample Code

```
{
                "subject": "Approve purchase of the new monitor for John Doe",
                "description": "John Doe has requested a new monitor, because
the old one has been broken",
                "priority": "MEDIUM",
                "dueDate": 20180217122851,
                "processor": "JaneDoe",
                "recipientUsers": "JaneDoe, AlexSmith",
                "recipientGroups": "Managers, HRs"
}
```

To remove a due date, recipient users, recipient groups, or the processor from a task, use an empty string:

Sample Code

```
{
                "dueDate": "",
                "processor": "",
                "recipientUsers": "",
                "recipientGroups": ""
}
```

Expressions

You can use Expressions [page 78] to refer to the context of the relevant workflow instance while updating the task properties, for example:

Sample Code

```
{
                "subject": "Approve purchase order for $
{context.employee.name} ${context.employee.surname}",
                "description": "Price: ${context.price*context.saleReduction}
EUR"
}
```

If the workflow instance context is as follows:

Sample Code

```
{
                "employee": {
```

```
                                "name": "John",
                                "surname": "Doe"
                },
                "price": 8000,
                "saleReduction": 0.5
}
```

then the task has the subject `Approve purchase order for John Doe` and the description `Price: 4000 EUR`.

Simultaneously Updating and Completing Tasks

With the same API endpoint that is used for updating the tasks you can also complete the tasks. See task patch API.

The *Workflow Participant* role must be assigned to your user. In addition, `"status": "COMPLETED"` and optionally `"context"` must be present in the payload, for example:

Sample Code

```
{
                "context": {
                                "price": 6000,
                                "reductionReason": "Outdated"
                },
                "status": "COMPLETED"
}
```

To update and complete the task with the same request, the *Workflow Administrator* role must be assigned to your user. The payload then looks, for example, as follows:

Sample Code

```
{
                "context": {
                                "price": 6000,
                                "reductionReason": "Outdated"
                },
                "status": "COMPLETED",
                "subject": "Approve purchase order for $
{context.employee.name} ${context.employee.surname}",
                "description": "Price: ${context.price*context.saleReduction}
EUR"
}
```

This has the following implications. First, the context of the relevant workflow instance is updated accordingly. Second, the task properties are updated considering the new values of the context. And, finally, the task status changes to `"COMPLETED"`.

In the above example, after the operation is performed, the subject of the task still is "Approve purchase order for John Doe", but the description is set considering the new values: "Price: 3000 EUR".

Related Information

Authorization Configuration [page 229]

3.3.8 Workflow Execution Log

The workflow execution log contains details about the execution history of a workflow instance.

The workflow execution log collects information that might be of use or interest to either a business user or an administrator. However, it isn't a technical log.

Logged Entries/Events

Log Entry/Event	Description
WORKFLOW_STARTED	Workflow instance started.
WORKFLOW_COMPLETED	Workflow instance completed.
WORKFLOW_CANCELED	Workflow instance canceled.
WORKFLOW_SUSPENDED	Workflow instance suspended.
WORKFLOW_CONTINUED	Workflow instance continued after processing was stopped due to an error.
WORKFLOW_RESUMED	Workflow instance resumed.
WORKFLOW_CONTEXT_OVERWRITTEN_BY_ADMIN	Context administrator completely overrode the workflow context.
WORKFLOW_ROLES_PATCHED_BY_ADMIN	Administrator changed the instance-specific role assignment of the given workflow instance.
WORKFLOW_CONTEXT_PATCHED_BY_ADMIN	Context administrator partially modified the workflow context.
USERTASK_CREATED	User task created.
USERTASK_CLAIMED	User task claimed.
USERTASK_COMPLETED	User task completed.
USERTASK_RELEASED	User task released.
USERTASK_PATCHED_BY_ADMIN	User task status, its properties, or its context was changed by administrator.
USERTASK_CANCELED_BY_BOUNDARY_EVENT	User task canceled by a boundary timer event.
USERTASK_FAILED	User task failed
SERVICETASK_CREATED	Service task created.
SERVICETASK_COMPLETED	Service task completed.

Log Entry/Event	Description
SERVICETASK_FAILED	Service task failed.
SCRIPTTASK_CREATED	Script task created.
SCRIPTTASK_COMPLETED	Script task completed.
SCRIPTTASK_FAILED	Script task failed.
MAILTASK_CREATED	Mail task created.
MAILTASK_COMPLETED	Mail task completed.
MAILTASK_FAILED	Mail task failed.
INTERMEDIATE_MESSAGE_EVENT_REACHED	Intermediate message event reached from a workflow instance.
INTERMEDIATE_MESSAGE_EVENT_TRIGGERED	Intermediate message event triggered for a workflow instance.
INTERMEDIATE_TIMER_EVENT_REACHED	Intermediate timer event reached in a workflow instance.
INTERMEDIATE_TIMER_EVENT_TRIGGERED	Intermediate timer event triggered for a workflow instance.
CANCELING_BOUNDARY_TIMER_EVENT_TRIGGERED	Boundary timer event triggered the cancellation of the attached user task and continued the alternative flow.
NONCANCELING_BOUNDARY_TIMER_EVENT_TRIGGERED	Boundary timer event triggered the alternative flow attached to it without canceling the attached user task.
EXCLUSIVE_GATEWAY_REACHED	Exclusive gateway reached in a workflow instance.
EXCLUSIVE_GATEWAY_FAILED	Exclusive gateway failed.
PARALLEL_GATEWAY_REACHED	Parallel gateway reached in a workflow instance.
PARALLEL_GATEWAY_FAILED	Parallel gateway failed.

3.3.9 Error Codes

If an error occurs while working with the SAP Workflow service API, the returned error object has an "errorCode" attribute.

This attribute identifies the area or workflow element where the problem occurred.

The table below describes the error code groups and points to the documentation that helps you fix the error.

Error Code / Error Code Prefix	Description	Related Links
`bpm.workflowruntime.generic.error`	This is a generic error. Contact SAP support citing the given log ID.	Monitoring and Troubleshooting [page 243]
`bpm.workflowruntime.expression`	There was a problem resolving an expression. This might be caused by the expression in the workflow definition or the data accessed through an expression, for example, the workflow context.	Expressions [page 78]
`bpm.workflowruntime.destination`	There was a problem resolving or accessing a destination.	Destinations [page 232] Configure the Workflow Service Mail Destination [page 177]
`bpm.workflowruntime.servicetask`	There was a problem executing a service task.	Configure Service Tasks [page 45]
`bpm.workflowruntime.mailtask`	There was a problem executing a mail task.	Configure Mail Tasks [page 60]
`bpm.workflowruntime.mailtask.connection`	There was a problem connecting to the mail server, either on network level or relating to the secure communication setup.	Configure Mail Tasks [page 60]
`bpm.workflowruntime.mailtask.server`	There was a problem while communicating with a mail server. The server refused the login or didn't accept a mail.	Configure Mail Tasks [page 60]
`bpm.workflowruntime.scripttask`	There was a problem executing a script task.	Configure Script Tasks [page 51]
`bpm.workflowruntime.usertask`	There was a problem executing a user task.	Configure User Tasks [page 29]
`bpm.workflowruntime.rest`	There was a problem calling the REST API.	Using Workflow APIs [page 151]

Related Information

API Hub: SAP Workflow Service Workflow

4 Consuming the Workflow Service at Application Runtime

There are tasks for the workflow service administrator and a user guide for business users of the workflow service.

Related Information

Configuring Workflow Service [page 169]
Using SAP Workflow Service [page 215]

4.1 Configuring Workflow Service

Before you can use the workflow service, meet the prerequisites and execute the basic setup.

Prerequisites

Initial Setup [page 17]

Procedure

1. Create a service instance of the SAP Workflow service.

   You can either create it in the cockpit or using the command-line interface:

   - Create a Service Instance of SAP Workflow Service Using the Cockpit [page 170]
   - Create a Service Instance of SAP Workflow Service Using the Command Line Interface [page 171]
2. (Optional) Create a service key for the service instance.

   For more information, see Create Service Keys Using the Cockpit in the SAP Business Technology Platform (SAP BTP) documentation.
3. Assign roles to your users.

   For more information, see Authorization Configuration [page 229], User and Member Management, and Assign Workflow Roles to Your Users [page 19].
4. Subscribe to your tool using the following steps:

- SAP Web IDE Full-Stack
  1. Navigate to your SAP BTP, Neo environment subaccount.
  2. In the navigation area, choose *Services*.
  3. Search and enable the SAP Web IDE Full-Stack.

  For more information, see Legacy: Enable the Workflow Editor in SAP Web IDE [page 86].
- SAP Business Application Studio
  1. Subscribe to the SAP Business Application Studio. For more information, see Subscribe to Multitenant Applications in the Cloud Foundry Environment Using the Cockpit.
     This enables the *Go to Application* link to access the application. You can then share the link with your developers.
  2. Assign the developers a role collection that contains the special role to access the application. For more information, see Manage Authorizations.
  3. Create a dev space in SAP Business Application Studio where you build your project and workflow module.
     For more information, see Create a Dev Space [page 20].

5. Configure SAP Fiori launchpad for the My Inbox and Monitor Workflows apps.

   For more information, see Create Workflow and My Inbox Tiles on Central Launchpad [page 184].

4.1.1 Create a Service Instance of SAP Workflow Service Using the Cockpit

You can create an instance of SAP Workflow service using either the cockpit or the command-line interface.

Context

In the SAP BTP, Cloud Foundry environment, you enable services by creating a service instance using either the SAP BTP cockpit or the SAP BTP, Cloud Foundry environment command line interface (CLI), and binding that instance to your application. A service instance is a single instantiation of a service running on SAP Business Technology Platform (SAP BTP). For more information, see About Services.

The service instance of workflow service scales dynamically according to the usage; any creation of additional service instances has no impact on the available resources. However, the creation of additional service instances might enable the integration with your applications in different spaces. All service instances within the same organization share the same data.

For technical authentication, we recommend that you create dedicated service instances to limit the number of authorizations granted to a single service instance. For more information, see Technical Authentication [page 232].

> ⚠ Caution
>
> As soon as you delete the last remaining service instance of the workflow service across all the spaces in your organization, all your data within the workflow service is irrevocably erased.

Procedure

1. Navigate to your space in the SAP BTP cockpit.
2. In the navigation area under *Services*, choose *Service Marketplace*.
3. Choose the *Workflow* tile.
4. Choose *Create*.
5. In the wizard, make sure that the service is set to *Workflow*, the plan ist set to *standard*, then enter a name for the instance, and choose *Next*.

   > **i** Note
   >
   > When you use a trial account, select the service plan *lite*.

6. (Optional) Provide a JSON object specifying a list of authorities to get granted to the OAuth client.

   For more information, see Technical Authentication [page 232].
7. Choose *Next*.
8. Check the overview, and then choose *Create*.

   The new instance appears on the *Instances and Subscriptions* page after a short while.
9. Create a service key for the service instance created.

   You need a service key if you want to call the service API standalone without a UI, for example, from Postman.

   a. On *Instances and Subscriptions* page, select the service instance and at the end of the row open the *Actions* menu (•••).
   b. Choose *Create Service Key*.
   c. In the *New Service Key* wizard, choose a name for your service key and provide configuration parameters either by uploading a JSON file or by configuring them in-line.

   For more information, see Create Service Keys Using the Cockpit in the SAP Business Technology Platform (SAP BTP) documentation.

4.1.2 Create a Service Instance of SAP Workflow Service Using the Command Line Interface

You can create the instance of SAP Workflow service using either the cockpit or the command-line interface.

Prerequisites

- Install the SAP BTP, Cloud Foundry environment Command Line Interface (CF CLI). For more information, see Download and Install the Cloud Foundry Command Line Interface.
- Log in to your organization and space using the CF CLI. For more information, see Creating Spaces Using the Cloud Foundry Command Line Interface.

Context

In the SAP BTP, Cloud Foundry environment, you enable services by creating a service instance using either the SAP BTP cockpit or the SAP BTP, Cloud Foundry environment command-line interface (CLI), and binding that instance to your application. A service instance is a single instantiation of a service running on SAP Business Technology Platform (SAP BTP). For more information, see About Services.

The service instance of workflow service scales dynamically according to the usage; any creation of additional service instances has no impact on the available resources. However, the creation of additional service instances might enable the integration with your applications in different spaces. All service instances within the same organization share the same data.

> ⚠ Caution
>
> As soon as you delete the last remaining service instance of the workflow service across all the spaces in your organization, all your data within the workflow service is irrevocably erased.

Procedure

1. Check that the workflow service is available on the marketplace:

   ```
   cf marketplace
   ```

   You should see a service named *workflow* with a plan named *standard*. If you use a trial account, the service plan is named *lite*.
2. Create a service instance of the workflow service:

   If you're using a global account, execute the following command:

   ```
   cf create-service workflow standard <instance-name>[-c <parameters-as-json>]
   ```

   If you're using a trial account, execute the following command:

   ```
   cf create-service workflow lite <instance-name>[-c <parameters-as-json>]
   ```

   Where `<instance-name>` is the service instance name of your choice. And where with the optional `-c` parameter the `<parameters-as-json>` is a JSON object (provided either inline or in a file) specifying a list of authorities to get granted to the OAuth client. For more information, see Technical Authentication [page 232].
3. (Optional) Bind the service instance to an application:

   ```
   cf bind-service <application> <instance-name>
   ```

   Where `cf bind-service <application>` is the name of a deployed application and `<instance-name>` is the service instance name used in step 2.

   For more information, see Binding Service Instances to Applications in the SAP Business Technology Platform (SAP BTP) documentation.

4. (Optional) Create a service key for the service instance:

   ```
   cf create-service-key workflow <instance-name> <service-key-name>
   ```

   Where `<instance-name>` is the service instance name used in step 2 and `<service-key-name>` is a name of your choice.

   For more information, see Create Service Keys Using the Cloud Foundry Command Line Interface in the SAP Business Technology Platform (SAP BTP) documentation.

4.1.2.1 Retrieve the Configuration of a Service Instance

You can determine the parameters of a service instance.

Procedure

Check the current configuration of a service instance of workflow service with the following commands:

```
cf service <instance-name> --guid
cf curl /v2/service_instances/<guid>/parameters
```

Where `<instance-name>` is the name of the service instance and `<guid>` is the output of the first command.

4.1.3 Configure the Access to Workflow Data

By default, all workflow service instances that are directly created in a subaccount as well as those created indirectly through subscription to an application that uses the workflow service share their content in this subaccount.

Context

When accessing data through the Rest API, there's a distinction between direct access to a single entity and queries for a collection of entities. You can always access single entities of the subaccount, independent from the service instance or subscription that was used to create it. For collection queries, only the entities of the current access path are returned by default (service instance or subscription). However, you can configure this behavior using the service instance parameter `defaultCollectionQueryFilter` and the respective value `shared` or `own` (default).

> **i** Note
>
> - Workflow data that originates from a subscription to an application is deleted once the application is unsubscribed.

- Applications that use the workflow service underneath can introduce an application scope. This scope defines a semantic bracket for the data that is related to the application. The application scope is configured with the `applicationScope` parameter. The required value is defined by the respective application.

Procedure

1. Prepare a JSON object that specifies which results to consider in queries that return a collection.

   The following example grants access to the shared workflow data:

   Sample Code

   ```
   {
       "defaultCollectionQueryFilter": "shared"
   }
   ```

2. Create or update a service instance for the SAP Workflow service and provide the JSON object.
   - Create a new service instance
     Provide the JSON object as parameter while creating a new service instance. See Create a Service Instance of SAP Workflow Service Using the Cockpit [page 170] and Create a Service Instance of SAP Workflow Service Using the Command Line Interface [page 171].
   - Update an existing service instance
     To add or remove authorities of an existing service instance, update the service instance with the JSON object. This is also possible for service instances, which were initially created without a JSON object. See Update Service Instances Using the Cloud Foundry Command Line Interface.

Related Information

Retrieve the Configuration of a Service Instance [page 173]

4.1.4 Enable Technical Authentication

Access the workflow service with a technical user.

Context

The authorities of a technical authentication refer to the scopes of the respective endpoints.

Procedure

1. To determine the list of scopes, you must identify the REST API endpoints that you want to invoke with the given OAuth2 client. To identify the scopes that are needed for a respective endpoint, see the SAP Workflow service API hub documentation. You can accumulate all required scopes in the authorities list.
2. Prepare a JSON object that specifies the list of authorizations you want to grant to the OAuth2 client that is provided through the service instance.

   The following example grants to the service instance the WORKFLOW_INSTANCE_START and MESSAGE_SEND authorities:

   Sample Code

   ```
   {
       "authorities": [
           "WORKFLOW_INSTANCE_START",
           "MESSAGE_SEND"
       ]
   }
   ```

3. Create or update a service instance for the SAP Workflow service and provide the JSON object.
   - Create a new service instance
     Provide the JSON object as parameter while creating a new service instance. See Create a Service Instance of SAP Workflow Service Using the Cockpit [page 170] and Create a Service Instance of SAP Workflow Service Using the Command Line Interface [page 171].
   - Update an existing service instance
     To add or remove authorities of an existing service instance, update the service instance with the JSON object. This is also possible for service instances, which were initially created without a JSON object. See Update Service Instances Using the Cloud Foundry Command Line Interface.

Results

You can use the resulting OAuth2 client with client credentials grant as provided by the service binding of the created service instance. For more information, see Access Workflow APIs Using OAuth 2.0 Authentication (Client Credentials Grant) [page 156].

Related Information

Technical Authentication [page 232]

4.1.5 Configuring Principal Propagation for Service Tasks

When starting a workflow or completing a task of a particular workflow instance, you can use principal propagation to forward the information about who is logged on to the services. The information is propagated throughout the workflow.

> **i** Note
>
> Technical authentication (see Technical Authentication [page 232]) isn't possible for operations that are configured to propagate the user to a subsequent service task. Principal propagation requires a user authenticated through an identity provider.

Before you can use principal propagation, the following one-time configurations are required:

- A destination that uses the OAuth credentials of the configuration parameters for the workflow service. For more information, see Create an OAuth Destination [page 176].

One destination for each service that is called. For more information, see Configure Service Tasks [page 45] and Destinations [page 232].

4.1.5.1 Create an OAuth Destination

To use principal propagation to forward the information about who is logged on to the services, you first need to create an OAuth destination.

Prerequisites

To configure destinations, use the standard SAP Business Technology Platform (SAP BTP) mechanisms in the SAP BTP cockpit. For more information, see Managing Destinations.

Log in to the cockpit and open the *Destinations* editor. See Access the Destinations Editor.

Procedure

Create an OAuth destination for the SAP Workflow service runtime using the following values:

Field	Value
Name	`bpmworkflowruntimeoauth`
Type	HTTP

Field	Value
Description	Enter a text, for example, **`SAP Workflow service OAuth Destination for Principal Propagation`**.
URL	Set the destination URL to the authorization endpoint URL found in the service key that was created for the workflow service. This corresponds to the URL in the `uaa.url` parameter as described in Determine Service Configuration Parameters [page 152].
Proxy	Internet
Authentication	Basic Authentication
User	Set the user to the client ID. See the `uaa.clientid` parameter as described in Determine Service Configuration Parameters [page 152].
Password	Set the password to the secret. See the `uaa.clientsecret` parameter as described in Determine Service Configuration Parameters [page 152]. ⚠ Caution Every service binding and service key can have their own client secret. If the binding or service key is deleted, the client secret can become invalid. Therefore, we recommend that you use the client secret from a dedicated service key that you don't plan to delete.

4.1.6 Configure the Workflow Service Mail Destination

Before you can send notification mails for mail tasks within a workflow, you must first configure a mail destination.

Prerequisites

- You have the details for configuring SMTP e-mail for your scenario.
- Your mail server has the following characteristics:
  - It supports the `SMTP STARTTLS` command on ports 587 or 465, because the workflow service supports only STARTTLS on these ports.
  - It requires authentication, because the workflow service doesn't support unauthenticated logins.
- In the SAP BTP, Cloud Foundry environment, you must have the following authorizations:
  Subaccount level: You must be a Global Account member to manage destinations and certificates (all create/read/update/delete operations) as well as to generate a keypair for trust management. See Add Global Account Members.

Also Security Administrators (which must be either Global Account members or SAP BTP, Cloud Foundry environment Organization/Space members) must be able to manage destinations at the subaccount level. See Managing Security Administrators in Your Subaccount [Feature Set A].

Procedure

Create a destination in the SAP BTP cockpit.

For more information, see Managing Destinations.

> **i Note**
>
> The proxy type *OnPremise* is not supported. That is, mail servers that can only be reached using Cloud Connector, are not supported.

- To import a destination:
  1. Save the template as a file.

```
Type=MAIL

Name=bpmworkflowruntime_mail

mail.user=
mail.password=

mail.smtp.host=mail.example.com
mail.smtp.port=587
mail.transport.protocol=smtp
mail.smtp.starttls.required=true
mail.smtp.starttls.enable=true
mail.smtp.auth=true

mail.smtp.from=cpworkflow@example.com
mail.smtp.ssl.checkserveridentity=true

mail.bpm.send.disabled=false
```

  2. Import the destination from the file, and set the values for user, password, host, port, and from address.
- To create a destination, use the following data and properties.

Field	Value
Name	**`bpmworkflowruntime_mail`**
Type	*Mail*
Description	Text that describes the destination, for example, **`Workflow service mail destination`**
User	User for logging in to the mail server

Field	Value
Password	Password for logging in to the mail server

Property	Value
`mail.transport.protocol`	**`smtp`**
`mail.smtp.auth`	**`true`**
`mail.smtp.starttls.required`	**`true`**
`mail.smtp.host`	Host name of your mail server
`mail.smtp.port`	Port on which your mail server listens for connections (typically **`587`**, in rare cases **`465`**)
`mail.smtp.from`	Mail address to use as the "From" address of mails sent by the workflow service, for example, **`cpworkflow@example.com`**. This address must belong to an existing mailbox because it receives the replies to mails that the workflow service sends.
`mail.smtp.ssl.checkserveridentity`	(Optional) **`true`** or **`false`**; default is **`true`** if no value is provided.
`mail.smtp.ssl.trust`	**`*`** or a space-separated list of acceptable host names. If you don't provide a value, trust is based on the certificate provided by the server, which must be part of the SAP JVM default truststore. For more information, see Determine the Trusted CAs of SAP JVM Truststore [page 180].
`mail.bpm.send.disabled`	○ **`true`** Turns off interaction with the mail server, for example, temporarily while you develop a workflow. ○ **`false`**

For more information about the properties, see the JavaMail API documentation.

> **i Note**
>
> Only the above properties are evaluated. Other properties that are configured in the mail destination have no effect. However, for an optimal operation of the mail functionality, the workflow service might apply additional properties, such as connection timeouts.

4.1.6.1 Determine the Trusted CAs of SAP JVM Truststore

With the information on the trusted certificate authorities, you can decide which certificate validation to use for mail server connections executed by mail tasks.

Context

You determine whether these connections can use the standard certificate validation or whether you need to override the validation using the trust list property `mail.smtp.ssl.trust`.

Workflow service runs with one of the latest SAP JVMs that are available. Therefore, the actual list can differ from what you determine locally, because trusted certificate authorities might get removed, for example, because their validity ends soon.

Procedure

1. Install a recent SAP JVM, for example, from SAP Development Tools CLOUD.
2. Determine the location of your installation. We assume, that your system runs on Linux and the JAVA_HOME environment variable is set to the location of your installation.
   a. Determine the location of the `cacerts` file of your installation. Typically, it's located relatively to the main folder of the SAP JVM at `$JAVA_HOME/jre/lib/security/cacerts`.
   b. Determine the location of the keytool program of your installation. Typically, it's located relatively to the main folder of the SAP JVM at `$JAVA_HOME/bin/keytool`.
3. Execute the following command line:

   ```
   "$JAVA_HOME/bin/keytool" -list -v -keystore "$JAVA_HOME/jre/lib/security/
   cacerts" > cacerts.list.txt
   ```

   a. Enter the keystore password when prompted. Initially, it usually is **`changeit`**.
4. Analyze the `cacerts.list.txt` file that is created in the current working directory and check whether the certificate chain of your server refers to one of the certificates listed.

4.1.7 Configure Document Management for Workflow Service Attachments

SAP Workflow service provides a lightweight integration with SAP Workflow Management. This means that binary files are completely be handled by SAP Workflow Management (storage, access management, and

more) while the workflow service maintains the relation between a workflow instance, its files, and additional metadata.

Prerequisites

- Set up a SAP Workflow Management service (integration option).
- Connect a compliant repository supporting CMIS 1.1 browser binding.
  For more information about SAP Document Management service and the setup, see SAP Document Management service.

Configure Attachments Destination

- We recommend that you configure a destination pointing to the configured repository, using the following attributes and values. See Managing Destinations.

Attribute	Value
Name	bpmworkflowruntime_attachments
Type	HTTP
URL	<repository-base-url>/browser/<repository-id>
Proxy Type	Internet
Authentication	OAuth2UserTokenExchange For more information, see OAuth User Token Exchange Authentication.
Client ID	<client ID>
Client Secret	<client secret>
Token Service URL	<uaa-url>/oauth/token
Token Service URL Type	Dedicated

- You can obtain client ID, client secret, and the token service URL from the *Service Key*.
  For more information, see https://docs.cloudfoundry.org/devguide/services/service-keys.html .
- Example service key: The relevant attributes are `uaa.clientid` (client ID), `uaa.clientsecret` (client secret) and `uaa.url` (token service URL):

Sample Code

```
{
     "uri": "https://api-sdm-di.cfapps.<region>.hana.ondemand.com/",
     "endpoints": {
```

```
        "ecmservice": {
            "url": "https://api-sdm-di.cfapps.<region>.hana.ondemand.com/",
            "timeout": 300000
        }
    },
    "sap.cloud.service": "com.sap.ecm.reuse",
    "saasregistryenabled": true,
    "html5-apps-repo": {
        "app_host_id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    },
    "uaa": {
        "uaadomain": "authentication.sap.hana.ondemand.com",
        "tenantmode": "dedicated",
        "sburl": "https://internal-
xsuaa.authentication.sap.<region>.ondemand.com",
        "clientid": "client-id",
        "verificationkey": "...",
        "apiurl": "https://api.authentication.sap.<region>.ondemand.com",
        "xsappname": "...",
        "identityzone": "...",
        "identityzoneid": "...",
        "clientsecret": "client-secret",
        "tenantid": "...",
        "url": "https://<org>.authentication.<region>.hana.ondemand.com"
    }
}
```

Related Information

Work with Attachments on a Workflow and Task Instance [page 160]

4.1.8 Configure an SAP API Business Hub Destination

Before you search for workflow APIs in SAP Business Application Studio, you must first configure a destination for SAP API Business Hub.

Prerequisites

- In the SAP BTP, Cloud Foundry environment, you must have the following authorizations:
  Subaccount level: You must be a Global Account member to manage destinations. See Add Global Account Members.

Procedure

Create a destination in the SAP BTP cockpit.

For more information, see Managing Destinations.

- To import a destination:
  1. Save the template as a `.txt` file.

     ```
     Description=SAP API Business Hub
     Type=HTTP
     TrustAll=true
     HTML5.DynamicDestination=true
     Authentication=NoAuthentication
     WebIDEUsage=apihub_catalog
     Name=SAP_API_Business_Hub
     WebIDEEnabled=true
     ProxyType=Internet
     URL=https\://api.sap.com\:443/
     ```

  2. Import the destination from the file.
- To create a destination, use the following data and properties.

Field	Value
Name	**SAP_API_Business_Hub**
Type	*HTTP*
Description	Text that describes the destination, for example, **Destinaton for SAP API Business Hub**
URL	**https://api.sap.com:443**
Proxy Type	Internet
Authentication	NoAuthentication

Property	Value
`HTML5.DynamicDestination`	**true**
`TrustAll`	**true**
`WebIDEEnabled`	**true**
`WebIDEUsage`	**apihub_catalog**

> **i Note**
>
> Only the above properties are evaluated. Other properties that are configured in the destination have no effect. However, for an optimal operation of the functionality, the workflow service might apply additional properties, such as connection timeouts.

4.1.9 Create Workflow and My Inbox Tiles on Central Launchpad

In the SAP BTP, Cloud Foundry environment, you can create *Monitor Workflows*, My Inbox, and *Start Form* tiles on SAP Fiori launchpad, using the SAP Launchpad service and its SAP Fiori launchpad functionalities.

Prerequisites

- You have subscribed the Launchpad application and configured the necessary roles for your user in your subaccount. See Initial Setup in the SAP Launchpad service documentation.
- You have a workflow service instance running. See Create a Service Instance of SAP Workflow Service Using the Cockpit [page 170].
- You have created a destination to the workflow service instance with the authentication method OAuth2JWTBearer. See Destinations Pointing to Service Instances.
- You have a site. See the Create a Site.

Context

> **i Note**
>
> Only workflows with custom task UIs that use the Managed Approuter can be used in the central launchpad. All other workflows need to use the legacy way for creating tiles. See Legacy: Create Workflow and My Inbox Tiles on SAP Fiori Launchpad [page 192].

Procedure

1. Navigate to your subaccount.
2. In the navigation area, open *Subscriptions* and access the *Launchpad* tile.
3. In the navigation area of the central SAP Fiori launchpad, choose *Provider Manager*.
4. To make the default *HTML5 Apps* content provider load the standard apps of the workflow service, choose the *Refresh* action.
5. In the navigation area, choose *Content Manager*, and then switch to the *Content Explorer* tab.
6. Choose the *HTML5 Apps* content provider, and then select the following items and choose *Add to My Content*:
    - My Inbox
    - Monitor Workflows
    - BPM Form Player
7. Switch to the *My Content* tab, and proceed with the following steps for each of your added Monitor Workflows, My Inbox, and BPM Form Player item:

a. Navigate into the item.
b. On the screen that opens, choose *Create a Local Copy*.
c. To use custom texts, choose *Edit* and adapt the texts in the *General* section.

   You can use a custom title, description, and subtitle for the locally created items. You can also adapt translations.

   > ⚠ Caution
   >
   > You must not change the *Configuration* data. Do not change the value in *SAPUI5 Component Name*. Do not change or delete the *subaccountID* or the *saasApprouter* configuration parameters.

d. Configure the locally created copies as described in the following sections:
   - My Inbox: Configure My Inbox App [page 185]
   - Monitor Workflows: Configure Monitor Workflows App [page 188]
   - BPM Form Player: Configure a Start-Form-Based Workflow Start App [page 188]

8. Save your changes.
9. Assign the created local copies of your items to a group and make sure that they are visible to users. See Assign Apps to a Group and to a Catalog in the SAP Launchpad service documentation.
10. Access the launchpad. See A Typical Workflow of Building a Site.

Results

Your tiles for the Monitor Workflows, My Inbox, and start-form-based workflow start apps are displayed.

4.1.9.1 Configure My Inbox App

Configuring the My Inbox app allows you to use its various functionalities in the *Master-Detail* and the *Expert View* layouts.

Prerequisites

You have created at least one local copy of the My Inbox app. See Create Workflow and My Inbox Tiles on Central Launchpad [page 184].

Context

Each display option needs an own local copy of the My Inbox app. The local copies of the items must be configured as follows.

Procedure

1. In the *Content Manager* of the central SAP Launchpad service, open the local copy of the My Inbox app.
2. Go to the *Navigation* tab of the local item, and choose *Edit*.
3. The default view of My Inbox is called *Master-Detail* view. To additionally enable the *Expert View* or other functionlalities, you have to manually set parameters for them.
   - Enable *Expert View* layout.

Parameter Name	Usage	Default Value	Permitted Values
`expertMode`	This parameter enables you to use the *Expert View* of My Inbox. For more information, see Expert View [page 219].	"false"	"false" or "true"

   - Show custom attributes.

Parameter Name	Usage	Default Value	Permitted Values
`showAdditionalAttributes`	The parameter enables you to display custom attributes corresponding to business-related data (custom attributes) that is assigned to your tasks. This will allow you to preview the data in a tabular representation, make use of advanced filter and search functionalities, and also sort and filter based on custom attributes. For more information, see Display Custom Attributes in My Inbox [page 41].	"false"	"false" or "true"

   - Enable substitutions.

Parameter Name	Usage	Default Value	Permitted Values
`substitution`	This parameter enables you to use the *Manage My Substitutes* option, displayed in the user action menu. You can manage your tasks in your	"true"	"false" or "true"

Parameter Name	Usage	Default Value	Permitted Values
	absence by creating substitution rules for planned and unplanned absences. For more information, see Create and Manage Substitution Rules [page 221]. Always use this parameter together with `userSearch`.		
`userSearch`	Enables the user search when creating a substitution rule or forwarding a task. The `userSearch` value must be set to "false".	"false"	"false"

- Enable forwarding.

Parameter Name	Usage	Default Value	Permitted Values
`userSearch`	The parameter `userSearch` allows you to forward a task to a user, if set to **`false`** and enabled for a user task. For more information, see Configure User Tasks [page 29].	"false"	"false"

- Limit the number of loaded tasks.

Parameter Name	Usage	Default Value	Permitted Values
`listSize`	Specify a numeric value to limit the number of tasks loaded in the My Inbox.	100	Integer

> ⚠ Caution
>
> For the proper usage of the My Inbox app, do not change the parameters on the *Navigation* tab, which are not listed above.

4. Save your changes.

4.1.9.2 Configure Monitor Workflows App

The Monitor Workflows app can display both workflow definitions as well workflow instances.

Prerequisites

You have created at least one local copy of the Monitor Workflows item. See Create Workflow and My Inbox Tiles on Central Launchpad [page 184].

Context

Each display option needs an own local copy of the Monitor Workflows item. The local copies of the items must be configured accordingly.

Procedure

1. In the *Content Manager* of the central SAP Fiori launchpad, open the local copy of the Monitor Workflows item.
2. Go to the *Navigation* tab of the local item, and then choose *Edit*.
3. Set one of the follow intent actions:
   - Display workflow definitions: `DisplayDefinitions`
   - Display workflow instances: `DisplayInstances`

> ⚠ Caution
>
> You must not change the preset *Semantic Object*.

4.1.9.3 Configure a Start-Form-Based Workflow Start App

To make a start form available to your end users, you must configure it as a tile.

Prerequisites

- You have created a start form. See Create Your Form [page 133].
- You have created at least one local copy of the BPM Form Player item. See Create Workflow and My Inbox Tiles on Central Launchpad [page 184].

Context

You always configure a generic form player application *BPM Form Player* that can interpret and render a start form. Using parameters you can adapt the relevant start form ID, its revision, and the form title. Each start form needs an own local copy of the BPM Form Player item. The items must be configured accordingly.

The following parameters are available and are configured in the subsequent steps:

- `formDefinitionId`: The ID of the start form you want to render.
- `revision`: The revision of the start form you want to render.
- `appTitle`: The application title that is rendered when you open the start form.
- `formTitle`: Optional. The title that is rendered in the form's header. If you do not set this parameter, no header is rendered.

Procedure

1. In the *Content Manager* of the central SAP Fiori launchpad, open the local copy of the BPM Form Player item.
2. Go to the *Navigation* tab of the local item, and then choose *Edit*.
3. Set the parameters by adding new parameters that match the ones of your start form configuration:

Name	Default Value
formDefinitionId	request-approval-form
revision	2.0 or Draft
appTitle	Request Approval
formTitle	Create Approval Request

4.1.9.4 Define Scenario-Specific Tiles in the Expert View of My Inbox

A scenario is an aggregation of predefined task types, exposed in a tile. In a scenario-specific tile the business users are able to work only on tasks, which are of specific, preconfigured task types. Whereas in the *All Items* tile, business users see all the tasks they are responsible for, regardless of the type of the tasks.

Prerequisites

You have created a local copy of the My Inbox app. See Create Workflow and My Inbox Tiles on Central Launchpad [page 184].

Context

With SAP Workflow service you can define scenario-specific tiles for My Inbox using the `taskDefinitions` app parameter, so that the business users in your organization work more efficiently with My Inbox. For more information, see Scenario-Specific Tiles [page 226].

Procedure

1. In the navigation area of the central SAP Launchpad service, choose *Content Manager* () and select your local copy of My Inbox. Fill in the following information:
   - Go to the *Navigation* tab and choose *Edit*. In the *Parameters* section, add `taskDefinitions` as parameter and a comma-separated list of `TaskDefinitionIDs` as value.

     > → Tip
     >
     > To get the **`TaskDefinitionID`**, follow the steps:
     >
     > 1. From the *All Items Tile* in My Inbox select the task, which you would like to see in the scenario-specific tile.
     > 2. Go to the user menu and navigate to *More* > *Support Information*, and capture the **`TaskDefinitionID`** value.

     For example:

Parameter	*Default Value*
`taskDefinitions`	**`usertask1@step1employeeonboarding,usertask2@step2employeeonboarding`**
`expertMode`	**`true`**

   - Go to the *Visualization* tab.
     - In the *General* section, add the subtitle of your workflow scenario in the *Subtitle* field, for example, **`Employee Onboarding related tasks`**.
     - In the *Dynamic Data section* > *OData Service URL* add the OData service URL extended with the task definitions.
       Without any filters applied, the OData service URL should look like as follows:

       ```
       /bpmworkflowruntime/odata/v1/tcm/TaskCollection/$count/?$filter=Status eq 'READY' or Status eq 'RESERVED' or Status eq 'IN_PROGRESS' or Status eq 'EXECUTED'
       ```

     - Extend the *OData Service URL* using **`TaskDefinitionID`** parameter as described in the following example:

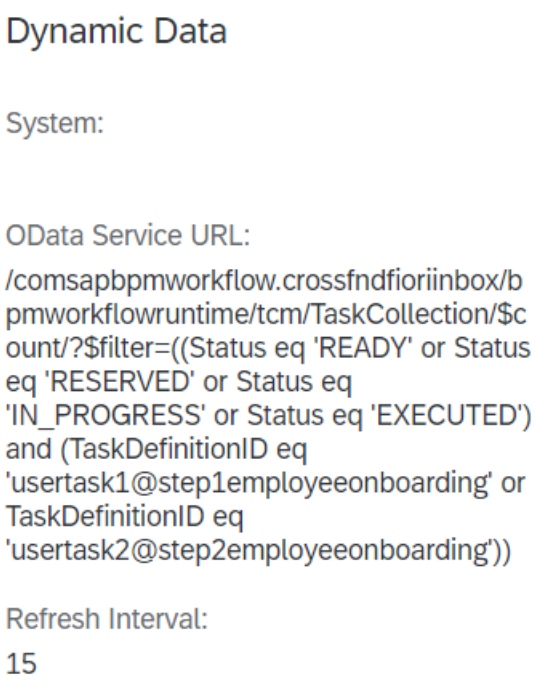
       ```
       /bpmworkflowruntime/odata/v1/tcm/TaskCollection/$count/?$filter=((Status eq 'READY' or Status eq 'RESERVED' or Status eq
       ```

```
'IN_PROGRESS' or Status eq 'EXECUTED') and (TaskDefinitionID eq
'usertask1@step1employeeonboarding' or TaskDefinitionID eq
'usertask2@step2employeeonboarding'))
```

> **i Note**
>
> The list of **`TaskDefinitionIDs`** in the *OData Service URL* newly added task-based filter must match the **`TaskDefinitionIDs`** in the **`TaskDefinitions`** parameter, so the tile counter of the scenario-specific tile to match the number of tasks displayed after the application is opened.

- *Icon:* Use any of the available icons.

After you edit the *OData Service URL* parameter, it should look as follows:



2. Save the changes.
3. To open your scenario-specific tile, go to *Site Directory* ( ) and choose *Go to site* ( ).

4.1.9.5 Legacy: Create Workflow and My Inbox Tiles on SAP Fiori Launchpad

In the SAP BTP, Cloud Foundry environment, you can create *Workflow Definitions*, *Workflow Instances*, and My Inbox tiles on SAP Fiori launchpad, using the SAP Cloud Portal service and its FLP functionalities. Follow the instructions to create a dedicated FLP module, to build and deploy it.

Context

> **i** Note
>
> This procedure only applies to existing customers that already use SAP Fiori launchpad modules. For new customer after **January 15, 2021, see** Create Workflow and My Inbox Tiles on Central Launchpad [page 184].

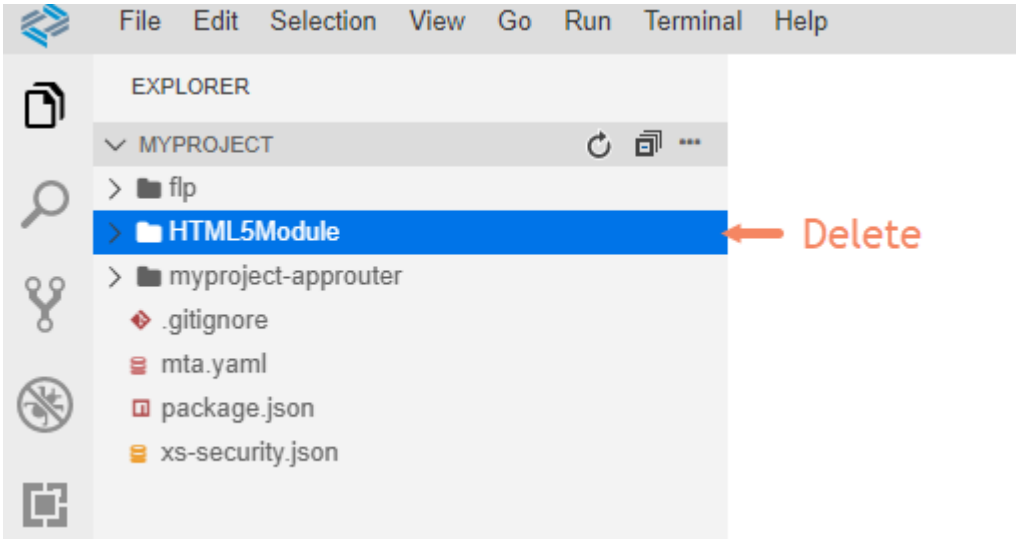
Using SAP Business Application Studio

Procedure

1. Create a multi-target application with the SAP Fiori launchpad module in the SAP BTP, Cloud Foundry environment. Follow the procedure described in Create a Multitarget Application with a Launchpad Module, and make sure you select the *SAP Fiori Freestyle Project* template.

   > **i** Note
   >
   > `<MTA_ID>` is further used as a placeholder for the name of your project. In the examples below, the value of `<MTA_ID>` is *MyProject*. For your project implementation, you always have to replace `<MTA_ID>` with the name of your project.

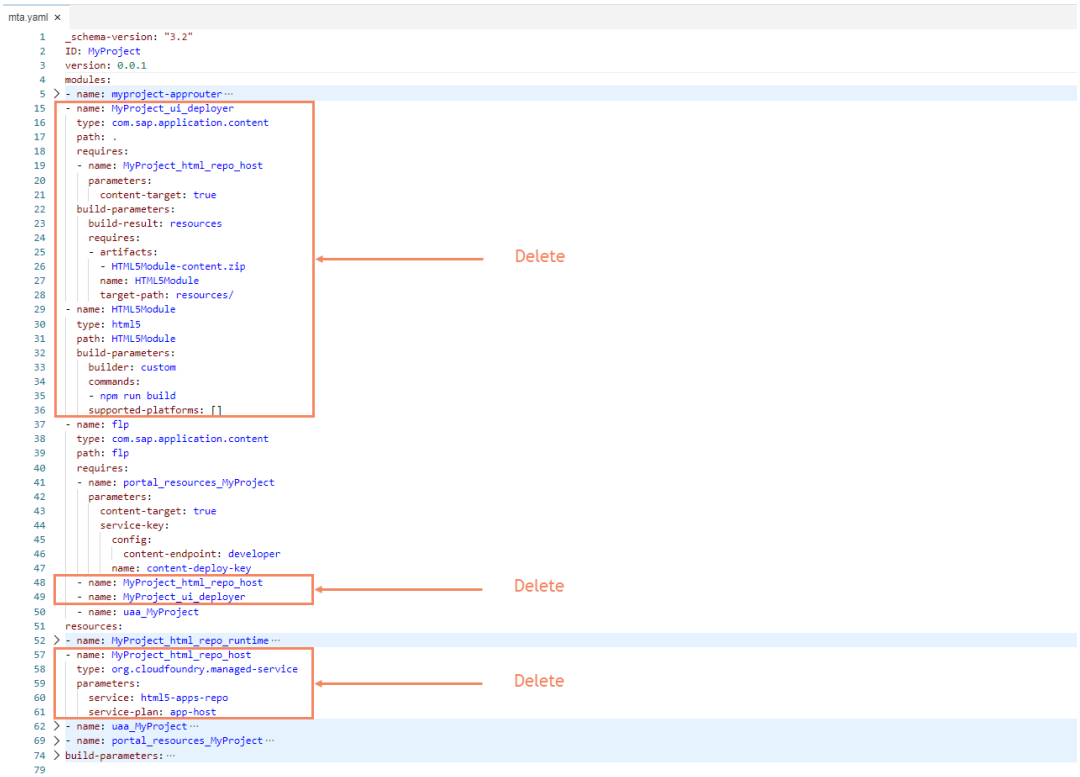
2. Delete the `HTML5Module` folder located in your project. The folder might be named differently depending on the name you have used in the previous step. Execute the action either by pressing the `Delete` keyboard button or by right-clicking the folder and choosing *Delete*.



3. Right-click the `mta.yaml` file located in your project and select *Open With* > *Code Editor*.

> ⚠ Caution
>
> While you work in the *Code Editor* view, make sure that the correct indentation is preserved and there are no trailing white spaces.

   - For SAP Fiori launchpad module:
     1. Delete the `<MTA ID>_ui_deployer` and `HTML5Module` from the `modules` section. The `HTML5Module` might be named differently depending on the name you have used in the first step. Delete the `<MTA ID>_html_repo_host` and `<MTA ID>_ui_deployer` service bindings from the `requires` section of the flp module. Also remove the `<MTA ID>_html_repo_host` resource from the `resources` section.



2. Add a dependency to the workflow service in the `requires` section of your project.

> **i Note**
>
> - Verify that you have provided correct indentations using spaces in the `mta.yaml` file.
> - `<workflow_service_instance_name>` is the workflow service instance name created in the cockpit.
> - `<FLP module name>` is the SAP Fiori launchpad module name created in SAP Business Application Studio.

```
- name: <FLP module name>
  type: com.sap.portal.content
  path: <FLP module path>
  parameters:
    stack: cflinuxfs3
    memory: 128M
    buildpack: https://github.com/cloudfoundry/nodejs-buildpack/
releases/download/v1.6.21/nodejs-buildpack-v1.6.21.zip
  requires:
    - name: portal_resources_<MTA ID>
    - name: <workflow_service_instance_name>
    - name: uaa_<MTA ID>
```

3. Add dependency to the workflow service instance in the `resources` section of your project:

```
- name: <workflow_service_instance_name>
  type: org.cloudfoundry.existing-service
```

- Add `uaa` dependency:
  1. If there's no `xs-security.json` file available, then create an `xs-security.json` file present at the same level as `mta.yaml`< file in the MTA folder. Add the following code, providing a unique `xsappname`:

```
{
  "xsappname":"<unique_xsapp_name>",
  "tenant-mode": "dedicated",
  "description": "Security profile of called application",
  "scopes": [
    {
      "name": "uaa.user",
      "description": "UAA"
    }
  ],
  "role-templates": [
    {
      "name": "Token_Exchange",
      "description": "UAA",
      "scope-references": [
        "uaa.user"
      ]
    }
  ]
}
```

  2. In the `mta.yaml` file, if not already available, add the following code under the `resources` section:

```
  - name: uaa_<MTA ID>
    parameters:
      path: ./xs-security.json
      service-plan: application
      service: xsuaa
    type: org.cloudfoundry.managed-service
```

- For the application router module, add dependency to the workflow service and the uaa service under the `requires` section of the application router module.

```
- name: <MTA ID>_appRouter
  type: approuter.nodejs
  path: <MTA ID>_appRouter
  parameters:
    disk-quota: 256M
    memory: 256M
  requires:
    - name: <MTA ID>_html5_repo_runtime
    - name: portal_resources_<MTA ID>
    - name: <workflow_service_instance_name>
    - name: uaa_<MTA ID>
```

4. In the `xs-app.json` file, inside `<MTA ID>_appRouter` folder of the application router, change the `authenticationMethod` to `route` by replacing the content of the file with the following code:

```
{
    "welcomeFile": "/cp.portal",
    "authenticationMethod": "route"
}
```

5. Add the tiles to SAP Fiori launchpad:
   - Open the `CommonDataModel.json` file located under the `portal-site` folder in the SAP Fiori launchpad site module with the *Launchpad Editor*.

- In the *Groups* tab, you can edit the *Default Group Title* or select an existing group and choose + to add a new tile.
- Provide the following details for `App ID` and `Intent Navigation`:

`App ID`	`Intent Navigation`
`cross.fnd.fiori.inbox`	`WorkflowTask-DisplayMyInbox`
`com.sap.bpm.monitorworkflow`	`bpmworkflowmonitor-DisplayInstances`
`com.sap.bpm.monitorworkflow`	`bpmworkflowmonitor-DisplayDefinitions`

- Choose *Select* to finish.

6. Build and deploy your project. For more information, see Build Your Application and Deploy Your Application.
7. Add the required roles to users. For more information, see Authorization Configuration [page 229].
8. Open the SAP Fiori launchpad.
   You can now see the tiles My Inbox, *Monitor Workflows (Workflow Instances)* and *Monitor Workflows (Workflow Definition)*.
9. (Optional) Customize your My Inbox app tile. For more information, see Configure My Inbox App [page 185].

Using SAP Web IDE

Prerequisites

1. Create a service instance of workflow service. For more information, see:
   - Create a Service Instance of SAP Workflow Service Using the Cockpit [page 170]
   - Create a Service Instance of SAP Workflow Service Using the Command Line Interface [page 171].
2. You must enable the workflow editor extension in SAP Web IDE Full-Stack. For more information, see Legacy: Enable the Workflow Editor in SAP Web IDE [page 86]
3. For paid accounts, your user has the entitlements for the following services assigned:
   - *SAP Cloud Portal service*
   - *Application Runtime*

   For more information, see Configure Entitlements and Quotas for Subaccounts.

Procedure

1. Log in to the SAP Web IDE Full-Stack application.
2. From the left pane, choose </> (Development) and navigate to the *Workspace* folder.
3. Choose *File* > *New Project from Sample Application* .
4. From the *Sample Application Selection* screen, choose *Category* as *Workflow Sample Application*.

5. Choose one of the following applications:
   - If you're a paid global account user, choose *Workflow Applications on SAP Fiori Launchpad for Cloud Foundry*
   - If you're a trial user, choose *Workflow Applications on SAP Fiori Launchpad for Cloud Foundry (Trial)*
6. Choose *Next*.
7. Select the checkbox and provide your consent to create the application.
8. Choose *Finish*.
9. (Optional) To use an existing instance of workflow service, define it by using `org.cloudfoundry.existing-service` resource type along with the existing resource name.

   In your workspace of the mta project, open the `mta.yaml` file. You can find the existing instance name of the workflow service by navigating to *Services* > *Service Instances* in the cockpit. Replace all occurrences of the existing instance name.

   Sample Code

   ```
   resources:
     - name: <existing_workflow_service_instance_name>
       type: org.cloudfoundry.existing-service
   ```

   Make sure to replace the instance name in the modules sections as well.
10. Build and deploy your project. For more information, see Packaging and Deploying Applications to Production Systems.
11. Add the required roles to users. For more information, see Authorization Configuration [page 229].
12. Access the SAP Fiori launchpad with the Monitor Workflows app tile, see Access Launchpad Runtime. You can now see the tiles with the titles *Workflow Instances*, *Workflow Definition*, and My Inbox on SAP Fiori launchpad.
13. (Optional) Customize your My Inbox app tile. For more information, see Configure My Inbox App [page 185].

4.1.9.5.1 Customize My Inbox

Use the *Expert View* layout or expose additional business-related data for your tasks, such as project name or project ID, in My Inbox.

Context

> **Note**
>
> This procedure only applies to existing customers that already use SAP Fiori launchpad modules. For new customer after January 15, 2021, see Configure My Inbox App [page 185].

You can customize the existing My Inbox tile of the *Master-Detail* view or add a new tile using personalization parameters that are supported by My Inbox. These parameters are defined in the Business App JSON file of

your multitarget application (MTA) project (with SAP Web IDE) SAP Fiori launchpad module (with SAP Business Application Studio). For more information, see Configuring Business Apps for Custom Visualization (with SAP Web IDE). The table below provides an overview of the personalization parameters supported by My Inbox:

Parameter Name	Usage	Default Value	Permitted Values
`expertMode`	Enables the *Expert View* of My Inbox.	"false"	"false" or "true"
`fullWidth`	Expands the application to the entire browser window.	"false"	“false” or “true”
`showAdditionalAttributes`	Displays business-related data (custom attributes).	"false"	"false" or "true"
`substitution`	Enables the *Manage My Substitutes* function in My Inbox.	"true"	"false" or "true"
`userSearch`	Enables the user search when creating a substitution rule or forwarding a task. The `userSearch` value must be explicitly set to "false".	"false"	"false" or "true"

The parameter `expertMode` enables you to use the *Expert View* of My Inbox. For more information, see Expert View [page 219].

The parameter `fullWidth` allows users to use their screen space more efficiently, by making the application fully responsive to the size of the screen.

The parameter `showAdditionalAttributes` enables you to display custom attributes corresponding to business-related data that is assigned to your tasks. This will allow you to preview the data in a tabular representation, make use of advanced filter and search functionalities, and also sort and filter based on custom attributes. For more information, see Display Custom Attributes in My Inbox [page 41].

The parameters `substitution` and `userSearch` enable you to use the *Manage My Substitutes* option, displayed in the user action menu. You can manage your tasks in your absence by creating substitution rules for planned and unplanned absences. For more information, see Create and Manage Substitution Rules [page 221].

> **i Note**
>
> If you don't see *Manage My Substitutes* in My Inbox by default, you have to redeploy the MTA project, as described in Create and Customize a New My Inbox Tile with SAP Web IDE [page 203].

Furthermore, the parameter `userSearch` allows you to forward a task to a user, if set to **`false`** and enabled for a user task. For more information, see Configure User Tasks [page 29].

You can enable the following features of My Inbox:

- Expert View [page 219] of My Inbox
- Display Custom Attributes in My Inbox [page 41]

Using this configurational approach, you have the following options:

- Create and Customize the Existing My Inbox Tile with SAP Business Application Studio [page 199]
- Create and Customize a New My Inbox Tile with SAP Web IDE [page 203]
- Create and Manage Substitution Rules [page 221]

4.1.9.5.1.1 Create and Customize the Existing My Inbox Tile with SAP Business Application Studio

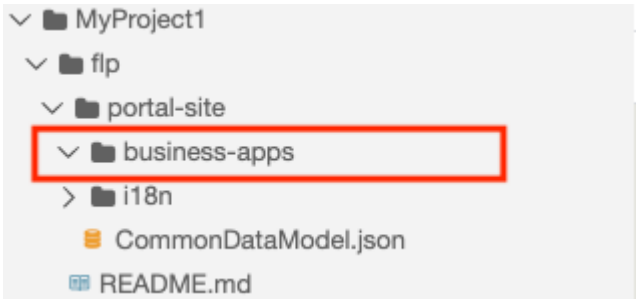
This procedure changes the *Master-Detail* layout of your existing My Inbox tile into the *Expert View* layout, which displays custom attributes that are configured during design time.

Context

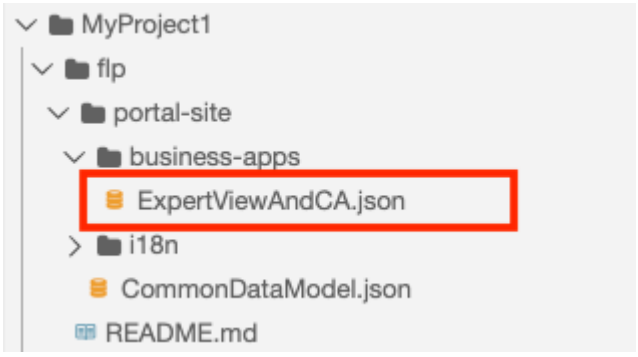
To create a new My Inbox tile, execute steps 1-8 described in Create Workflow and My Inbox Tiles on Central Launchpad [page 184] and continue with the procedure below to customize the tile.

Procedure

1. Create a folder called `business-apps` in the `portal-site` folder of the SAP Fiori launchpad module.



2. Create a JSON file within the `business-apps` folder and give it a name.



3. Import the following content into the created JSON file, depending on the setup you require:

```
{
```

```
    "_version":"3.0",
    "identification":{
        "id":"cross.fnd.fiori.inbox",
        "entityType":"businessapp"
    },
    "payload":{
        "visualizations":{
            "WorkflowTask-DisplayMyInboxExpertMode":{
                "vizType":"sap.ushell.DynamicAppLauncher",
                "vizConfig":{
                    "sap.app":{
                        "title":"Expert mode with custom attributes",
                        "description":"DisplayMyInboxExpertModeDesk",
                        "tags":{

                        }
                    },
                    "sap.flp":{
                        "target":{
                            "inboundId":"WorkflowTask-DisplayMyInbox",
                            "parameters":{
                                "expertMode":{
                                    "value":{
                                        "value":"true",
                                        "format":"plain"
                                    }
                                },
                                "showAdditionalAttributes":{
                                    "value":{
                                        "value":"true",
                                        "format":"plain"
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

In this example, you set the parameters `expertMode` and `showAdditionalAttributes` to `"true"` in order to activate respectively the expert mode or the custom attributes. These parameters are not interdependent. Depending on your requirement, you can set only one or both of them. For more information about custom attributes, see Configure Custom Task Attributes [page 40] and Display Custom Attributes in My Inbox [page 41].

- If you don't want to activate the `expertMode`, make sure you remove the following from the JSON file:

```
"expertMode":{
    "value":{
        "value":"true",
        "format":"plain"
    }
},
```

- If you don't want to activate the custom attributes, make sure you remove the following from the JSON file:

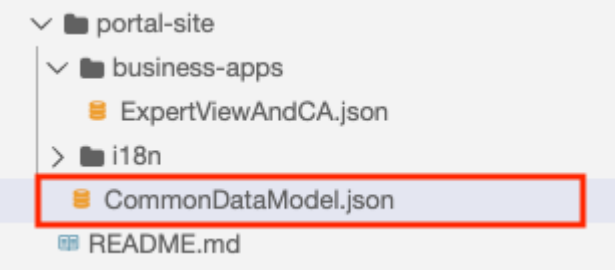
```
"showAdditionalAttributes":{
    "value":{
        "value":"true",
        "format":"plain"
    }
```

```
},
```

4. Add a new My Inbox tile on the SAP Fiori launchpad.

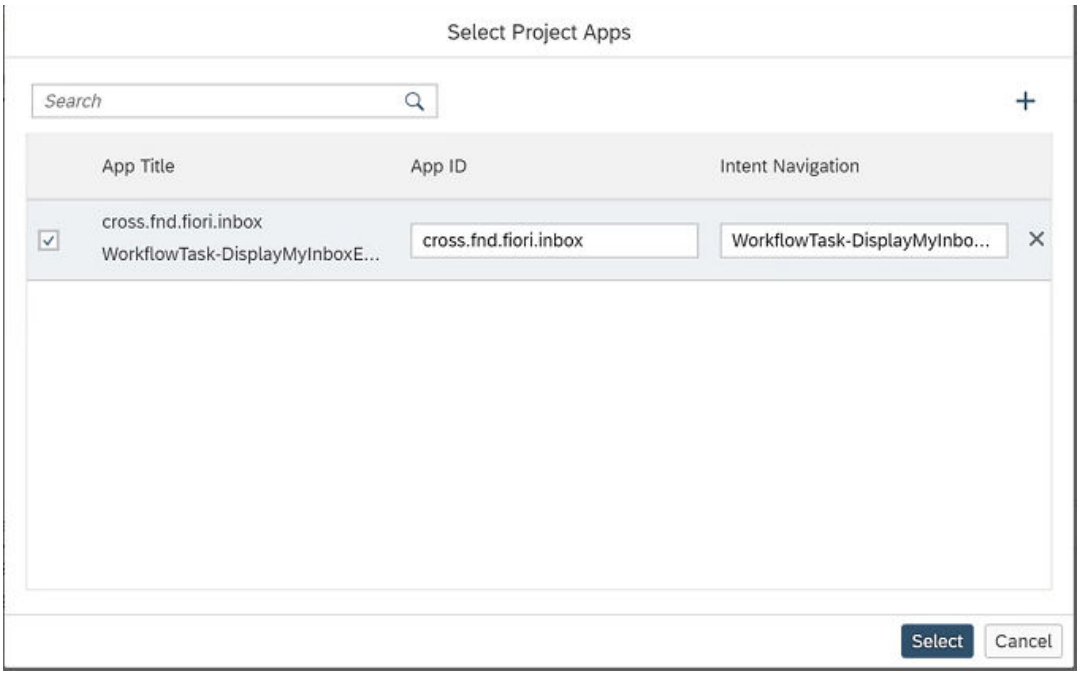
   You can create and customize a new My Inbox tile on the SAP Fiori launchpad by using the *Launchpad Editor* or the *Code Editor*:

   1. Using the *Launchpad Editor*:
      - In the *Launchpad Editor* under *SAP Fiori launchpad module* > *portal-site* folder, open the `CommonDataModel.json` file.

      - In the *Groups* tab, you can edit the *Default Group Title* or select an existing group and choose +.
      - In the top-right corner of the *Select Project Apps* window, choose + to add a new tile.
      - Provide the following details for *App ID* and *Intent Navigation*:

App ID	Intent Navigation
`cross.fnd.fiori.inbox`	`WorkflowTask-DisplayMyInboxExpertMode`

      - Choose *Select* to finish.

2. Using the *Code editor*:
   - In the *Launchpad Editor* under *SAP Fiori launchpad module* > *portal-site* folder, open the `CommonDataModel.json` file.
   - Add a reference to the expert mode as a node in the payload for "groups":

```
"groups":[
    {
        "_version":"3.0.0",
        "identification":{
            "id":"defaultGroupId",
            "title":"{{title}}",
            "entityType":"group",
            "i18n":"i18n/defaultGroupId.properties"
        },
        "payload":{
            "viz":[
                {
                    "id":"com.sap.bpm.monitorworkflow-0-1551956290611",
                    "appId":"com.sap.bpm.monitorworkflow",
                    "vizId":"bpmworkflowmonitor-DisplayInstances"
                },
                {
                    "id":"com.sap.bpm.monitorworkflow-1-1551956290611",
                    "appId":"com.sap.bpm.monitorworkflow",
                    "vizId":"bpmworkflowmonitor-DisplayDefinitions"
                },
                {
                    "id":"cross.fnd.fiori.inbox-2-1551956290611",
                    "appId":"cross.fnd.fiori.inbox",
                    "vizId":"WorkflowTask-DisplayMyInboxExpertMode"
                },
                {
                    "id":"cross.fnd.fiori.inbox-3-1557918418298",
                    "appId":"cross.fnd.fiori.inbox",
                    "vizId":"WorkflowTask-DisplayMyInbox"
                }
            ]
        }
    }
]
```

> **Note**
>
> The `visualization ID` (`vizId`) of the My Inbox tile in `CommonDataModel.json` from step 4b must correspond to the `visualization` key defined in the business-apps JSON file, created in step 3. In this case, `"WorkflowTask-DisplayMyInboxExpertMode"`.

5. (Optional) Customize the My Inbox tile.

   To customize the existing My Inbox tile, you should change the `<Intent Navigation>` of the tile to the defined `visualization` key (for example, `"WorkflowTask-DisplayMyInboxExpertMode"`) in the business-apps JSON file.

   To do this, execute the following steps in the *Code editor*:

   - Open the `CommonDataModel.json` file, located under the `'portal-site'` folder in the SAP Fiori launchpad site module.
   - Add a reference to the expert mode as a node in the payload for `"groups"`, as shown below:

```
"groups":[
   {
```

```
      "_version":"3.0.0",
      "identification":{
         "id":"defaultGroupId",
         "title":"{{title}}",
         "entityType":"group",
         "i18n":"i18n/defaultGroupId.properties"
      },
      "payload":{
         "viz":[
            {
               "id":"com.sap.bpm.monitorworkflow-0-1551956290611",
               "appId":"com.sap.bpm.monitorworkflow",
               "vizId":"bpmworkflowmonitor-DisplayInstances"
            },
            {
               "id":"com.sap.bpm.monitorworkflow-1-1551956290611",
               "appId":"com.sap.bpm.monitorworkflow",
               "vizId":"bpmworkflowmonitor-DisplayDefinitions"
            },
            {
               "id":"cross.fnd.fiori.inbox-2-1551956290611",
               "appId":"cross.fnd.fiori.inbox",
               "vizId":"WorkflowTask-DisplayMyInboxExpertMode"
            }
         ]
      }
   }
]
```

> **i** Note
>
> The `visualization ID` (`vizId`) of the My Inbox tile in `CommonDataModel.json` from step 4 must correspond to the `visualization` key defined in the business-apps JSON file, created in step 3. In this case, `"WorkflowTask-DisplayMyInboxExpertMode"`.

6. Build and deploy your MTA project. For more information, see Build Your Application and Deploy Your Application.

4.1.9.5.1.2 Create and Customize a New My Inbox Tile with SAP Web IDE

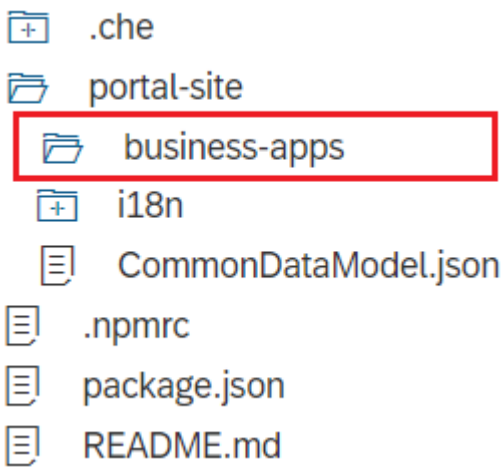
Follow this procedure to create a new My Inbox tile with the *Expert View* layout that will display custom attributes.

Context

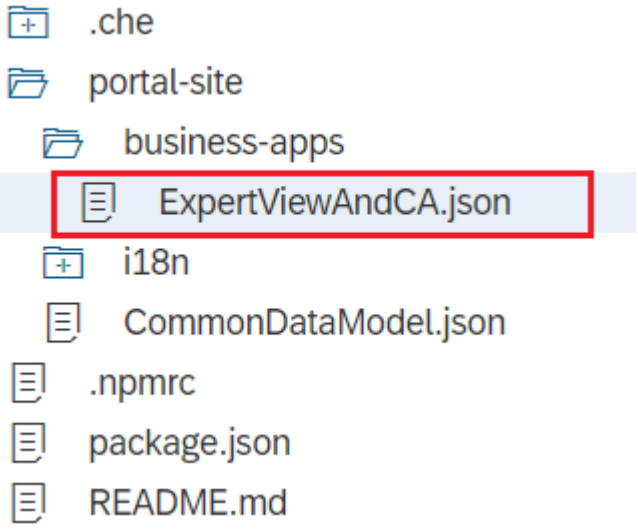
To create a new My Inbox tile, execute steps 1-8 described in Create Workflow and My Inbox Tiles on Central Launchpad [page 184] and continue with the procedure below to customize the tile.

Procedure

1. Create a folder called `business-apps` in the `portal-site` folder of the SAP Fiori launchpad module.



2. Create a JSON file within the `business-apps` folder and give it a name.



3. Import the following content into the created JSON file, depending on the setup you require:

```
{
    "_version":"3.0",
    "identification":{
        "id":"cross.fnd.fiori.inbox",
        "entityType":"businessapp"
    },
    "payload":{
        "visualizations":{
            "WorkflowTask-DisplayMyInboxExpertMode":{
                "vizType":"sap.ushell.DynamicAppLauncher",
                "vizConfig":{
                    "sap.app":{
                        "title":"Expert mode with custom attributes",
                        "description":"DisplayMyInboxExpertModeDesk",
                        "tags":{

                        }
```

```
                    },
                    "sap.flp":{
                        "target":{
                            "inboundId":"WorkflowTask-DisplayMyInbox",
                            "parameters":{
                                "expertMode":{
                                    "value":{
                                        "value":"true",
                                        "format":"plain"
                                    }
                                },
                                "showAdditionalAttributes":{
                                    "value":{
                                        "value":"true",
                                        "format":"plain"
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

In this example, you set the parameters `expertMode` and `showAdditionalAttributes` to `"true"` in order to activate respectively the expert mode or the custom attributes. These parameters are not interdependent. Depending on your requirement, you can set only one or both of them. For more information about custom attributes, see Configure Custom Task Attributes [page 40] and Display Custom Attributes in My Inbox [page 41].

- If you don't want to activate the `expertMode`, make sure you remove the following from the JSON file:

```
"expertMode":{
    "value":{
        "value":"true",
        "format":"plain"
    }
},
```

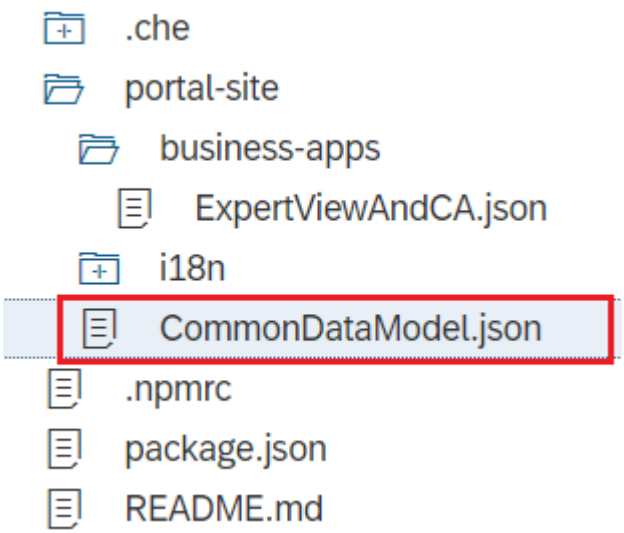
- If you don't want to activate the custom attributes, make sure you remove the following from the JSON file:

```
"showAdditionalAttributes":{
    "value":{
        "value":"true",
        "format":"plain"
    }
},
```

4. Add a new My Inbox tile on the SAP Fiori launchpad.

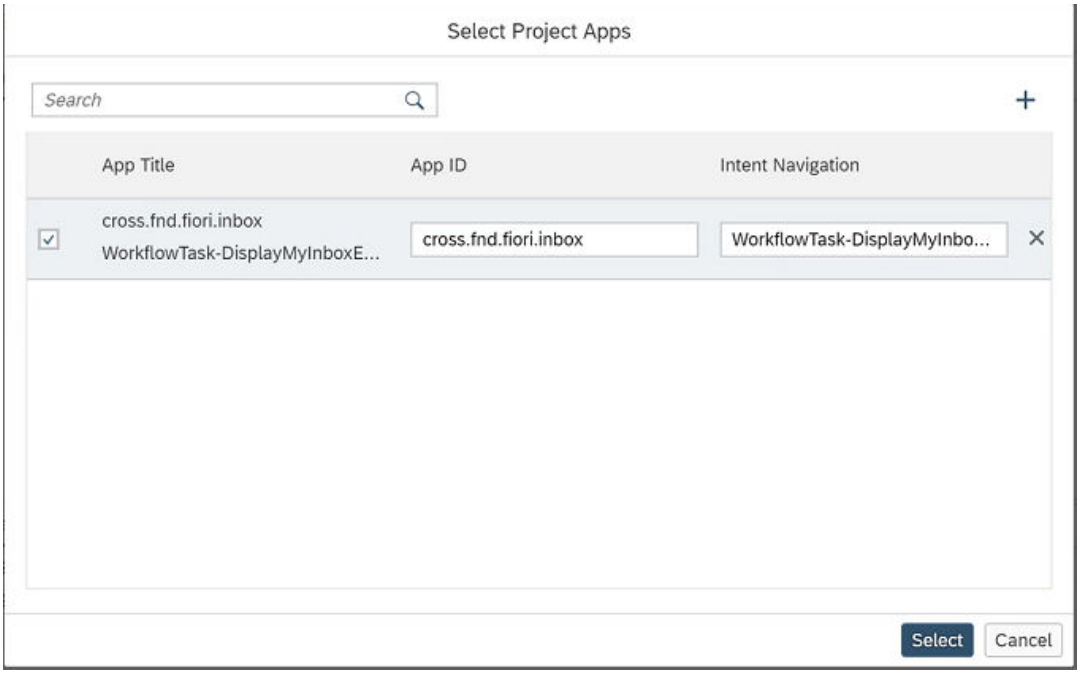
   You can create and customize a new My Inbox tile on the SAP Fiori launchpad by using the *Launchpad Editor* (SAP Web IDE) or the *Code Editor*:

   1. Using the *Launchpad Editor*:
      - In the *Launchpad Editor* under *SAP Fiori launchpad module* > *portal-site* > *folder* >, open the `CommonDataModel.json` file.



- In the *Group Tile* tab, choose ✎ to edit the *Default Group Title* and choose + to add tiles.
- Add the tile by choosing + in the top-right corner.
- Provide the following details for *App ID* and *Intent Navigation*:

App ID	Intent Navigation
`cross.fnd.fiori.inbox`	`WorkflowTask-DisplayMyInboxExpertMode`



- Choose *Select* to finish.

2. Using the *Code editor*:

- In the *Launchpad Editor* under *SAP Fiori launchpad module* > *portal-site* > *folder* >, open the `CommonDataModel.json` file.
- Add a reference to the expert mode as a node in the payload for "groups":

```
"groups":[
    {
        "_version":"3.0.0",
        "identification":{
            "id":"defaultGroupId",
            "title":"{{title}}",
            "entityType":"group",
            "i18n":"i18n/defaultGroupId.properties"
        },
        "payload":{
            "viz":[
                {
                    "id":"com.sap.bpm.monitorworkflow-0-1551956290611",
                    "appId":"com.sap.bpm.monitorworkflow",
                    "vizId":"bpmworkflowmonitor-DisplayInstances"
                },
                {
                    "id":"com.sap.bpm.monitorworkflow-1-1551956290611",
                    "appId":"com.sap.bpm.monitorworkflow",
                    "vizId":"bpmworkflowmonitor-DisplayDefinitions"
                },
                {
                    "id":"cross.fnd.fiori.inbox-2-1551956290611",
                    "appId":"cross.fnd.fiori.inbox",
                    "vizId":"WorkflowTask-DisplayMyInboxExpertMode"
                },
                {
                    "id":"cross.fnd.fiori.inbox-3-1557918418298",
                    "appId":"cross.fnd.fiori.inbox",
                    "vizId":"WorkflowTask-DisplayMyInbox"
                }
            ]
        }
    }
]
```

> **i Note**
>
> The `visualization ID` (`vizId`) of the My Inbox tile in `CommonDataModel.json` from step 4b must correspond to the `visualization` key defined in the business-apps JSON file, created in step 3. In this case, `"WorkflowTask-DisplayMyInboxExpertMode"`.

5. Build and deploy your MTA project.

4.1.9.5.2 Configure a Tile for a Start Form on SAP Fiori Launchpad with SAP Business Application Studio

To make a start form available to your end users, you must integrate it into your portal and or SAP Fiori launchpad and then configure a custom tile.

Context

> **i** Note
>
> This procedure only applies to existing subaccounts that use SAP Fiori launchpad modules. **If you have a subaccount that was created after January 15, 2021, see** Configure a Start-Form-Based Workflow Start App [page 188].

Legacy Procedure

Prerequisites

- Either create a new SAP Fiori project that contains the SAP Fiori launchpad module, or use the existing one created for the Workflow tiles (recommended). See Create Workflow and My Inbox Tiles on Central Launchpad [page 184].
- You've created a start form. See Create Your Form [page 133].

Context

You can configure custom tiles in SAP Fiori launchpad using business application visualizations. For more information, see Adding Custom Visualization.

Procedure

1. Configure the business application visualization for your custom tile that is located in `<name of your SAP Fiori Launchpad module> / portal-site / business-apps`), for example, `app.json`. If you don't have a *business-apps* folder, create it. Use the following template:

```
{
  "_version": "3.0.0",
  "identification": {
    "id": "com.sap.bpm.wus.form.player",
    "entityType": "businessapp",
```

```
    "i18n": "i18n/<translation-file-name>.properties"
  },
  "payload": {
    "visualizations": {
      "<custom-id>": {
        "vizType": "sap.ushell.StaticAppLauncher",
        "vizConfig": {
          "sap.app": {
            "title": "{{title}}",
            "subTitle": "{{subtitle}}"
          },
          "sap.flp": {
            "target": {
              "inboundId": "bpmworkflow-Start",
              "parameters": {
                "formDefinitionId": {
                  "value": {
                    "value": "<form-definition-id>",
                    "format": "plain"
                  }
                },
                "revision": {
                  "value": {
                    "value": "<revision>",
                    "format": "plain"
                  }
                },
                "appTitle": {
                  "value": {
                    "value": "{{apptitle}}",
                    "format": "plain"
                  }
                },
                "formTitle": {
                  "value": {
                    "value": "{{formtitle}}",
                    "format": "plain"
                  }
                }
              }
            }
          }
        }
      }
    }
  }}
```

Technically, you always configure a generic form player application that can interpret and render a start form. Using the parameters you can adapt the relevant start form ID, its revision, and the form title.

The following parameters are available and are configured in the subsequent steps:

- `formDefinitionId`: The ID of the start form you want to render.
- `revision`: The revision of the start form you want to render.
- `appTitle`: The application title that is rendered when you open the start form. It's translatable following the translation concept of SAP Fiori launchpad.
- `formTitle`: Optional. The title that is rendered in the form's header. It's translatable following the translation concept of SAP Fiori launchpad. If not set, no header is rendered.
- title: The title to be displayed on your custom tile.
- subTitle: The subtitle to be displayed on your custom tile.

2. Replace `<form-definition-id>` and `<revision>` in your `app.json`. Use the values defined for the start form that you want to provide to your end users in the SAP Fiori launchpad. You can omit the `<custom-id>` property for the moment. It's configured in **Step 5**.

Parameter	Sample Value
`<form-definition-id>`	**`request-approval-form`**
`<revision>`	**`2.0`** or **`Draft`**

3. For translation, create the corresponding properties file that is located in `<name of your SAP Fiori Launchpad module> / portal-site / i18n`).

   For example, `app.properties` referring to the variables in your `app.json`. Use the following template:

```
#XTIT
title=<your tile title>
#XTIT
subtitle=<your tile subtitle>
#XTIT
apptitle=<your app title>
#XTIT
formtitle=<your form title>
```

   Example:

Parameter	Sample Value
`title`	**`Request Approval`**
`subtitle`	**`Self-Service`**
`apptitle`	**`Request Approval`**
`formtitle`	**`Create Approval Request`**

> **i Note**
>
> Follow the naming conventions for translation files. See Handling Translation.

4. Reference the newly created properties file within your `app.json`. To do so, replace the `<translation-file-name>.properties` parameter with the actual file name, for example, `app.properties`.
5. Set a custom ID for the custom configuration that you created. To do so, replace the parameter `<custom-id>`, for example, `request-approval-app-config`.
6. Refer to that custom configuration in the `CommonDataModel.json` of your SAP Fiori launchpad module. To do so, use the `<custom-id>` you specified in the configuration of the business application visualization, for example, `request-approval-app-config`.

   The relevant snippet looks like this:

```
...
"groups": [
```

```
  "..."
  {
    "_version": "3.0.0",
    "identification": {
      "id": "defaultGroupId",
      "title": "{{title}}",
      "entityType": "group",
      "i18n": "i18n/defaultGroupId.properties"
    },
    "payload": {
      "viz": [
      "..."
        {
          "id": "appid-0-1556008257548",
          "appId": "com.sap.bpm.wus.form.player",
          "vizId": "<custom-id>"
        }
      ]
    }
  }
]
...
```

7. Build and deploy your project. For more information, see Build Your Application and Deploy Your Application.

Results

In your SAP Fiori launchpad, your custom tile renders with the parameters that you specified. Users who select the tile navigate to the generic form player application, which renders the configured start form.

4.1.10 Create Custom Start UI Tiles on the Central Launchpad

In the SAP BTP, Cloud Foundry environment, you can create tiles for your custom start UIs on SAP Fiori launchpad, using the SAP Launchpad service and its SAP Fiori launchpad functionalities.

Prerequisites

- You have subscribed the Launchpad application and configured the necessary roles for your user in your subaccount. See Initial Setup in the SAP Launchpad service documentation.
- You have a workflow service instance running. See Create a Service Instance of SAP Workflow Service Using the Cockpit [page 170].
- You have a SAP Fiori launchpad site. See the Create a Site.
- You have created a custom start UI. See Creating a Custom Start UI [page 120].

Context

> **i Note**
>
> Only workflows with custom task UIs that use the Managed Approuter can be used in the central SAP Fiori launchpad. All other workflows need to use the legacy way for creating tiles. See Legacy: Creating a Custom Start UI [page 126].

Procedure

1. Navigate to your subaccount.
2. In the navigation area, open *Subscriptions*.
3. On the *Launchpad* tile, choose *Go to Application*.
4. In the navigation area of the central SAP Fiori launchpad, choose *Provider Manager* ( ).
5. To make the default *HTML5 Apps* content provider load the standard apps of the workflow service, choose *Content Providers* > *HTML 5 Apps* and then the *Refresh* ( ) action.
6. In the navigation area, choose *Content Manager*, ( ), and then switch to the *Content Explorer* tab.
7. Select the HTML5 application representing your custom start UI and choose *Add to My Content* (+).
8. Switch to the *My Content* tab, and proceed with the following steps:
   a. Navigate into your added HTML5 application item.
   b. On the screen that opens, choose *Create a Local Copy*.
   c. To adapt the texts in the *General* section, choose *Edit*.

      You can use a custom title, description, and subtitle for the locally created item. You can also adapt translations.

      > **⚠ Caution**
      >
      > You must not change the *Configuration* data. Do not change the value in *SAPUI5 Component Name*. Do not change or delete the *subaccountID* or the *saasApprouter* configuration parameters.

   d. Switch to the *Navigation* tab and define an arbitrary intent.

      See this sample entry:

      Semantic Object: `MyWorkflowApp`

      Action: `Start`
   e. Switch to the *Visualization* tab, then choose a visualization type for the resulting tile of your item, for example, a static app launcher.
9. Save your changes.
10. Assign the created local copies of your item to a group and make sure that they are visible to users. See Assign Apps to a Group and to a Catalog in the SAP Launchpad service documentation.
11. Access the launchpad. See A Typical Workflow of Building a Site.

Results

The tile for your custom task UI is displayed.

4.1.11 Export SAP Workflow Service Data

The export provides access to your business data stored within the workflow. You can use this data to address, for example, audit needs.

Prerequisites

You have the *WorkflowTenantOperator* role that allows you to export runtime data related to workflow definitions, form definitions, workflow instances, and task instances.

Context

> ⚠ Caution
>
> The export doesn't contain technical details that are required to reimport the data to the workflow service.

You can export the following types of data from the workflow service:

- Design time or modeling artifacts from the workflow editor.
  For more information, see Transport Workflows between Accounts [page 74] in the SAP BTP, Cloud Foundry environment.
- Runtime artifacts from the workflow service using the SAP Workflow service API.
  You export data related to a workflow definition, form definition, a workflow instance, or a task instance.
  The exported data and format is based on the SAP Workflow service REST APIs, see Using Workflow APIs [page 151].

The following procedure describes the export of the runtime artifacts.

Procedure

To export the data, enter the following URL: **`<base URL>/v1/export`**.

You can find the base URL in Determine the Service Host [page 157].

Results

> ⚠ Caution
>
> To verify that the export completed successfully, please check that you can extract the zip archive. The archive shouldn't contain a file named `error-log.txt`. If there's an `error-log.txt` file, the exported data might be corrupt. Check the file for details.

The export call returns a zip file that contains the following:

- A `readme.txt` file that contains meta information about this specific export.
- A `form-definitions.json` file that contains a list of the latest deployed form definitions.
- A `workflow-definitions.json` file that contains a list of the latest deployed workflow definitions.
- A `workflow-instances.json` file that contains a list of all workflow instances available on the system. The custom workflow attributes are contained in each entry of the list.
- A `workflow-instance-data` folder: For each workflow instance on the system one file (`<workflow-instance-ID>.json`) is written. It contains the latest details related to this instance including, for example, context data, attachment information, and the execution log.
- A `task-instances.json` file that contains a list of all task instances available on the system. The custom task attributes are contained in each entry of the list.
- A `form-definition-data` folder: For each form definition on the system one file (`<form-definition-ID>.json`) is written. It contains form definition metadata of all versions deployed.
- A `substitution-rules.json` file that contains the list of substitution rules. The data has the following format. Note that more fields may be added over time.

Field	Type	Description
id	String	The substitution rule ID. The ID is at most 64 characters long.
substitutedUser	String	The name of the user who is substituted. The user ID is at most 255 characters long.
substitute	String	The name of the user who substitutes 'substitutedUser'. The user ID is at most 255 characters long.
createdAt	String	Format: date-time The time when the substitution rule was created.
beginDate	String	Format: date-time The time from when the substitution rule is active.
endDate	String	Format: date-time The time up to which the substitution rule is active.

Field	Type	Description
mode	String	Enumeration type: ○ RECEIVE_TASKS for planned substitutions ○ TAKE_OVER for unplanned substitutions
enabled	Boolean	Indicates whether the substitution rule is active. A rule of mode RECEIVE_TASKS is enabled by the substituted user, for example, when creating the rule due to a planned absence. A rule of mode TAKE_OVER is enabled by the substituting user, for example, when taking over tasks due to an unplanned absence of the substituted user.

4.1.12 Deactivate the Workflow Service

Administrators of the SAP Business Technology Platform (SAP BTP) account can disable the SAP Workflow service.

Context

> ⚠ Caution
>
> When you delete the last workflow service instance, all data from your subaccount is deleted.

Procedure

Delete the workflow service instances.

For more information, see Deleting Service Instances.

4.2 Using SAP Workflow Service

The user guide for the workflow service is for end-users and key-users.

End-users find information on Working with Tasks in My Inbox [page 216]:

- My Inbox Layout [page 218]

- Expert View [page 219]

Related Information

Managing Workflows Using the Monitor Workflows App [page 243]

4.2.1 Working with Tasks in My Inbox

You can process workflow service tasks within the My Inbox application, which runs on the SAP Fiori launchpad. You can use My Inbox on your desktop or mobile device.

Prerequisites

- Configure SAP Fiori launchpad objects. For more information, see Create Workflow and My Inbox Tiles on Central Launchpad [page 184].

A user task is a type of flow object that appears in My Inbox. You can work on a task, complete a task instance, and view its description.

My Inbox displays the following information about the workflow and tasks:

- Task title
- Tasks with status *Ready* and *Reserved*
- Tasks with priority

Key Features

- View the tasks that are assigned to you.
- Claim tasks.

  > **i Note**
  >
  > When you claim a task, you become its processor and its other recipients no longer see it in My Inbox. The status of a claimed task changes from *Ready* to *Reserved*.

- View your current and available tasks.
- Display the task count on the My Inbox tile in the SAP Fiori launchpad.
- Sort tasks by priority, due date, task title, and the user who created the task.
- Filter your tasks by priority, due date, status *Ready*, and creation date. The task list is limited to the first 1000 entries that match the filter. In the *Filtered by* header, you can view information about all the applied filters on the task list in a tooltip, which appears when you hover over the *Filter* with your mouse.

- Group tasks by task title, priority, status, and by task type. The task type is the name of the user task as it was assigned in the workflow model defined in the editor.
- View task-specific details.
- Release tasks for which you are the processor.

  > **i Note**
  >
  > When you release a task, you are no longer assigned as a processor of this task and it becomes visible in My Inbox for its other recipients. The status of the task changes from *Reserved* to *Ready*.

- View *Workflow Log*.
- Forward tasks. You can forward tasks to assign them to other users for processing. To enable forwarding, see Configure User Tasks [page 29].
  To retrieve the expected user ID format, select a task in My Inbox, open the user action menu and select *Support Information*. The value of the *CreatedBy* property is in the expected user ID format, for example *username@sap.com*.
- Execute and complete tasks.

  > **i Note**
  >
  > When you select a task from the *List* view, the task details are displayed in the *Details* view. The custom action buttons, added by following the procedure described in Add Task Completion Buttons to My Inbox [page 112], appear at the bottom of the screen. When you select one of the buttons, a *Custom Action Dialog* appears on your screen. You have the option to add a Decision Note. The field with label *Decision Note* is optional except for the case when it is marked with an asterisk (*) before it. In case the field should be filled in, the *Decision Option* button is active only if this requirement is fulfilled. Otherwise you are able to submit your decision without adding a Decision Note.

Rate Limits

To ensure optimal operation of the service, SAP Workflow service execution is subject to resource limits, for example, regarding the number of requests per second. If the limit is exceeded, My Inbox displays the following error message: `" Your action could not be performed because the usage limits are reached. Please retry later or contact your help desk for assistance.”`. The client should then reduce the number of calls. For more information, see Conventions, Restrictions, and Limits [page 8].

Related Information

Destinations [page 232]

4.2.1.1 My Inbox Layout

Depending on your use-case scenario, you can use the *Master-Detail* view or the *Expert View* of My Inbox for displaying and processing your tasks.

Master-Detail View

The *Master-Detail* view offers options for scanning, selecting, and navigating the tasks that are shown in the *Details area*.

In the *Master-Detail* view of My Inbox, you can perform search, filtering, sorting and grouping operations. It is optimized for mobile devices.

It allows you to perform all standard actions as follows:

- View details about the workflow for a selected task and events, relevant to it chronologically.

  > **i Note**
  >
  > SAP Workflow service shows only the user ID; it does not show additional user details.

- Claim a task to reserve it for processing.
- Release a task for which you are the processor.
- Share a task via e-mail.
- You can sort, group, and filter tasks.
- You can view workflow related information.
- Process tasks using custom actions.
  For more information, see Add Task Completion Buttons to My Inbox [page 112].

Expert View

The *Expert* view is a tabular representation of the standard attributes of your tasks in My Inbox.

It allows you to:

- View large number of tasks.
- See the standard attributes for a task, such as *Task Title*, *Status*, and *Priority*.
- Use an advanced custom search filter to find the tasks you want to process.
- Sort, group, and filter tasks.
- Personalize the view per user, and share with other users.

4.2.1.2 Expert View

The *Expert View* is a tabular representation of the standard attributes of your tasks in My Inbox.

> **i Note**
>
> The *Expert View* is disabled by default in My Inbox. To enable it, your administrator has to configure the additional parameter `expertMode=true` in the app configuration of My Inbox.

Expert View

Use the *Expert View* to perform any of the following tasks:

- View a large number of tasks.
- See the standard attributes for a task, such as *Task Title*, *Status*, and *Priority*.
- Quickly narrow the list down to tasks that you want to process by using the advanced custom search filter.
- Sort and group tasks.
- Personalize the view per user, and share it with other users.

> **i Note**
>
> The changes that you make to the *Expert View* of My Inbox are persistent. You can use the personalized My Inbox in another browser or device without having to make the changes again.

Open Tasks

In the *Expert View*, you can open a task by clicking the line item of the task.

With the default UI of My Inbox, the *Detail View* screen provides an information tab. If general custom attributes have been defined, they are shown here.

> **i Note**
>
> Your predefined custom attributes are shown in the header area of the *Task Details* screen. For more information, see Display Custom Attributes in My Inbox [page 41].

> **i Note**
>
> The *Created By* column of the *Expert View* contains empty values. To hide it, choose *Personalize* and deselect it from the *Columns* dialog. This action does not persist.

Show Log

To view details about a task workflow and its events chronologically, choose *Show Log*.

- Workflow Log
  The *Workflow Log* tab contains details about the workflow of a selected task and events relevant to it chronologically.

Claim

To reserve a task for processing, choose *Claim*.

> **i** Note
>
> When you claim a task, you become the processor of the task and all other recipients no longer see it in My Inbox. In this case, the status of the task changes from *Ready* to *Reserved*.

Release

You can release any task for which you are the processor.

> **i** Note
>
> Once you release a task, you are no longer assigned as one of its processors, and it becomes visible in My Inbox for its other recipients. The status of the task changes from *Reserved* to *Ready*.

4.2.1.2.1 Expert View Standard Operations

In the *Expert* view, you can search, filter, refresh, sort, and group tasks that require action. You can also personalize the table columns.

Search Tasks

Search all tasks by entering one or more keywords in the *Search* field.

> **i** Note
>
> The search operation is performed on the client side, that is, only among the tasks that are loaded into the UI.

Filter Tasks

Use the *Show Filter Bar* button to filter tasks by the following criteria: *Task Title*, *Status*, *Priority*, *Due By*, and *Created Within*.

Refresh Tasks

Use the *Refresh* function to update your table with tasks.

Sort Tasks

Use the *Sort* function to sort tasks on the following criteria: *Ascending*, *Descending*, *Task Title*, *Status*, *Priority*, *Due On*, and *Created On*.

Group Tasks

Use the *Group* function of the *Expert* view to group tasks by *Ascending*, *Descending*, *Task Type*, *Status*, *Priority*, *Due On*, *Created On*, *None*.

Personalize Table

To choose which columns (*All, Task Title, Status, Priority, Created By, Created On, Due Date*) appear in your table with tasks, use the *Personalize* function of the *Expert* view.

> **i** Note
>
> When you select a *Task*, the footer of the screen shows only the available standard task actions, for example, *Show Log*, *Claim*, *Release* or *Forward*. Custom task actions are shown when you open the *Task Details* view.

4.2.1.3 Create and Manage Substitution Rules

You can use My Inbox to create substitution rules to manage your tasks in your absence. The substitution rules can be created for planned and unplanned absences.

At runtime, you can use the respective workflow service API or Inbox API to search for custom task attributes or to find the respective task instances. For more information about the characteristics of the various APIs, see Using Workflow APIs [page 151].

The audit log data stored for your account is deleted on a regular basis. In case you want to retain and use the data after the defined period you can retrieve it via the Audit Log Retrieval API Usage for the Cloud Foundry Environment and store it in another persistent storage. For more information, see Audit Log Retention for the Cloud Foundry Environment.

Prerequisites

You have the *Manage My Substitutes* option, displayed in the user action menu. If you don't see this option, you have to redeploy the MTA project, as described in Create and Customize a New My Inbox Tile with SAP Web IDE [page 203].

Planned Substitution

Planned substitution is usually targeted for a scenario where you know the start date and the end date for your absence. Your substitute will then see your tasks directly displayed in their inbox for the period defined by you.

To create a substitution rule for a planned absence, proceed as follows:

1. In My Inbox press the user action menu and select *Manage My Substitutes*.
2. Make sure you have selected the *Planned* tab and choose *Add New Substitute* in the footer of the screen.
3. In the *Add Substitute* dialog, enter the user ID of the substitute you want to nominate.
   To retrieve the expected user ID format, select a task in My Inbox, open the user action menu and select *Support Information*. The value of the *CreatedBy* property is in the expected user ID format, for example *username@sap.com*.
4. Choose a period for the substitution and choose *Save*.

   > **i Note**
   >
   > If you do not select a substitution period and save the rule, it is planned from the day of the creation of the rule for an undefined period.

   The planned substitution rules are automatically activated on the start date you have selected, and are automatically deactivated on the end date, respectively. On the start date of the substitution rule, your substitute will receive the tasks you have defined in the substitution rule automatically. On the end date of the substitution rule, your substitute will stop receiving the tasks you have defined in the substitution rule automatically.
   Tasks, which have been already claimed by the substitute prior to the end date will stay in the substitute's inbox.

As a result, the successful creation of a planned nominee is confirmed and you can see the entry in the *Planned* substitution tab.

After you have created the new substitution rule, make sure that the user ID of your substitute is spelled correctly in the list of planned substitutions.

You will see *Active*, or *Inactive* status for each substitution rule you have created.

Unplanned Substitution

To create a substitution rule for an unplanned absence, proceed as follows:

1. In My Inbox press the user icon and select *Manage My Substitutes*.
2. Select the *Unplanned* tab.
3. Choose *Add New Substitute* in the footer of the screen.
4. In the *Add Substitute* dialog, enter the user ID of the substitute you want to nominate.
   To retrieve the expected user ID format, select a task in My Inbox, open the user action menu and select *Support Information*. The value of the *CreatedBy* property is in the expected user ID format, for example *username@sap.com*.
5. Choose *Save*.

   > **i Note**
   >
   > If you do not select a substitution period and save the rule, it is planned from the day of the creation of the rule for an undefined period.

   Tasks, which have been already claimed by the substitute prior to the end date will stay in the substitute's inbox.
   As a result, the successful creation of an unplanned nominee is confirmed and you can see the entry in the *Unplanned* substitution tab.
   After you have created the new substitution rule, make sure that the user ID of your substitute is spelled correctly in the list of unplanned substitutions.

   > **i Note**
   >
   > In this case, your substitute will need to accept the substitution in order to see your tasks in their inbox.

Take Over Tasks as an Unplanned Substitute

You can take over or stop receiving tasks from users, who have nominated you as their unplanned substitute.

To do so, proceed as follows:

1. In My Inbox press the user icon and select *Substitute For*.
   You will see the list of users who have nominated you as their unplanned substitute. By default, you will not be receiving their tasks. If you want to take over, you will need to activate the substitution for that particular user.
2. Click on the switch button to activate the substitution rule for the selected user.
3. After you activate or deactivate the user, choose *Done*.

If you deactivate the substitution rule, you will stop receiving tasks from the selected user.

4.2.1.4 Custom Attributes in My Inbox

The custom attributes in My Inbox show additional information about a task, depending on the task contextual data. For example, Project ID or Project Name. These additional details about tasks allow you to make more informed decisions about tasks while working with My Inbox.

Context

The custom attributes in My Inbox are supported both by the *Master-Detail View* and the *Expert View*.

> **i Note**
>
> The custom attributes need to be configured in advance so that they are displayed by My Inbox, see Display Custom Attributes in My Inbox [page 41].

Benefit

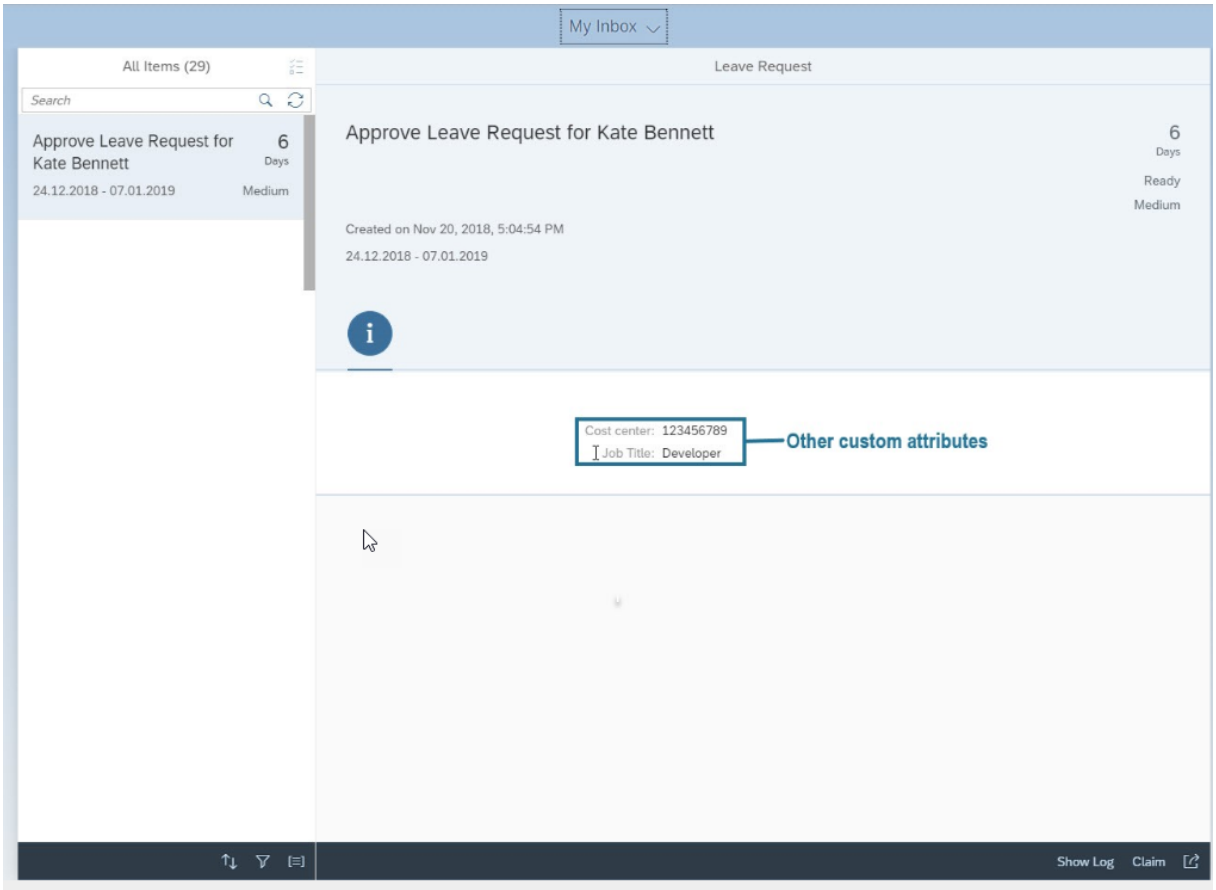
- View additional business data related to tasks to make more informed decisions.
- Quickly find tasks using sorting and filtering in the *Expert View*.

Use

Master-Detail View

The additional Information displayed by custom attributes is visualized on specific places on the My Inbox *Master-Detail View*.

My Inbox displays up to 3 KPI indicators in the task list and replicated in the header of the generic UI for task details. In addition, more business data about tasks is displayed in the *Information tab* of the generic UI for task details, following right after the description of the task. Refer to the screenshot below:

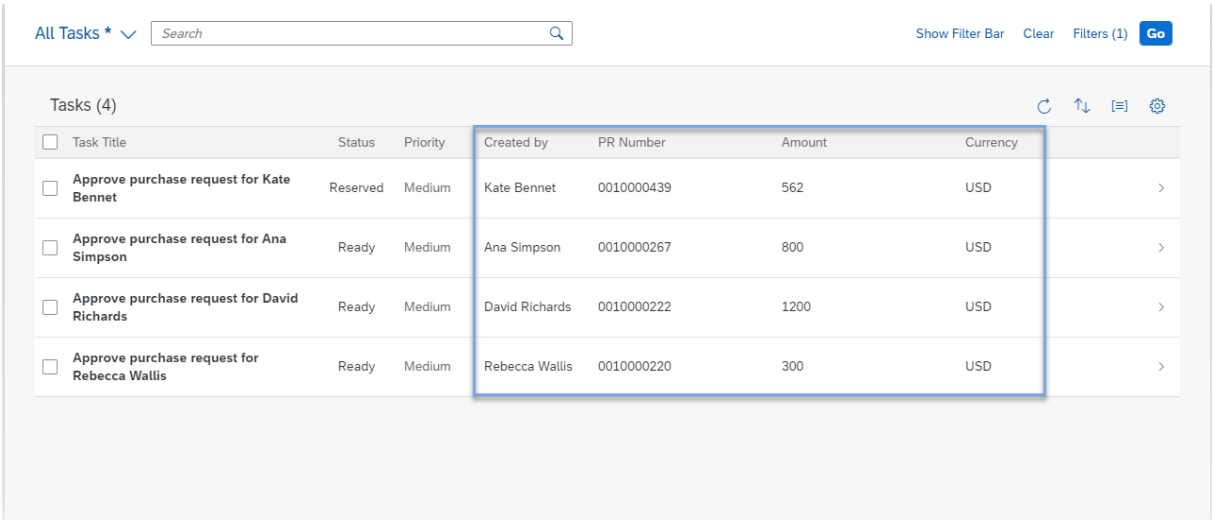


> **→ Tip**
>
> If you are working with a task showing a custom task UI, these custom attributes are not displayed in the task details page of My Inbox. In this case, you can use the respective Workflow API or Task Consumption Model API to search for custom attributes or to find the respective task instances. For more information about the APIs, see Using Workflow APIs [page 151].

> **i Note**
>
> Sorting and filtering on custom attribute data is not available with the *Master-Details View* of My Inbox. If you would like to sort and filter by custom attributes, consider using the *Expert View* of My Inbox.

Expert View

The custom attribute data about tasks can be exposed as columns in the *Expert View* of My Inbox. In addition, the custom attribute data is shown in the generic UI for task details. Refer to the screenshot below:



To display the custom attribute data available for tasks, do the following:

> **i Note**
>
> In order to see the available custom attributes in the *Expert View*, you first need to filter tasks by *Task Type*.

- Choose *Filters* button to open the *Filters* dialog
- Select at least one *Task Type* from the drop-down menu.
- (Optional) To filter the task list by a custom attribute, use the value help to provide a value for the selected field.
- (Optional) To display a search box in the *Filter Bar* of the *Expert View* for a given entry in the *Filters Dialog*, select the *Show on Filter Bar* checkbox for the selected entry.
- Choose *Go* button to apply your changes.
- The custom attributes, configured for the chosen Task Types, will be added as columns in the table.

> **Note**
>
> You can personalize the display of columns using the table *Personalize* button  . For more information, see Expert View Standard Operations [page 220].

4.2.1.5 Scenario-Specific Tiles

Scenario-specific tiles display a filtered set of tasks for domain-specific approvals in My Inbox. The feature is supported by the *Expert* view layout of My Inbox.

This feature is explicitly configured by an administrator in your organization. For more information, see: Define Scenario-Specific Tiles in the Expert View of My Inbox [page 189]

Scenario-Specific Tiles in Expert View

The *Expert* view-specific feature allows you to:

- Use the full set of functionalities offered by the *Expert* view.
- View the additional business data (custom attributes) rendered automatically as columns, as part of the of the *Expert View* table, without having to filter by task type.
- Sort or filter the tasks based on additional business data (custom attributes). For more information, see: Display Custom Attributes in My Inbox [page 41].

  > **Note**
  >
  > After the initial loading of My Inbox in *Expert View*, the usage of scenario-specific tile is marked by the *Filters (1)* value in the header of the *Table* view. This indicates, that the task list is prefiltered according to the scenario configuration.

  > **Note**
  >
  > You can personalize the display of columns using the table *Personalize* button(). For more information, see Expert View Standard Operations [page 220].

5 Security

This guide provides an overview of the security-relevant information that applies to the SAP Workflow service.

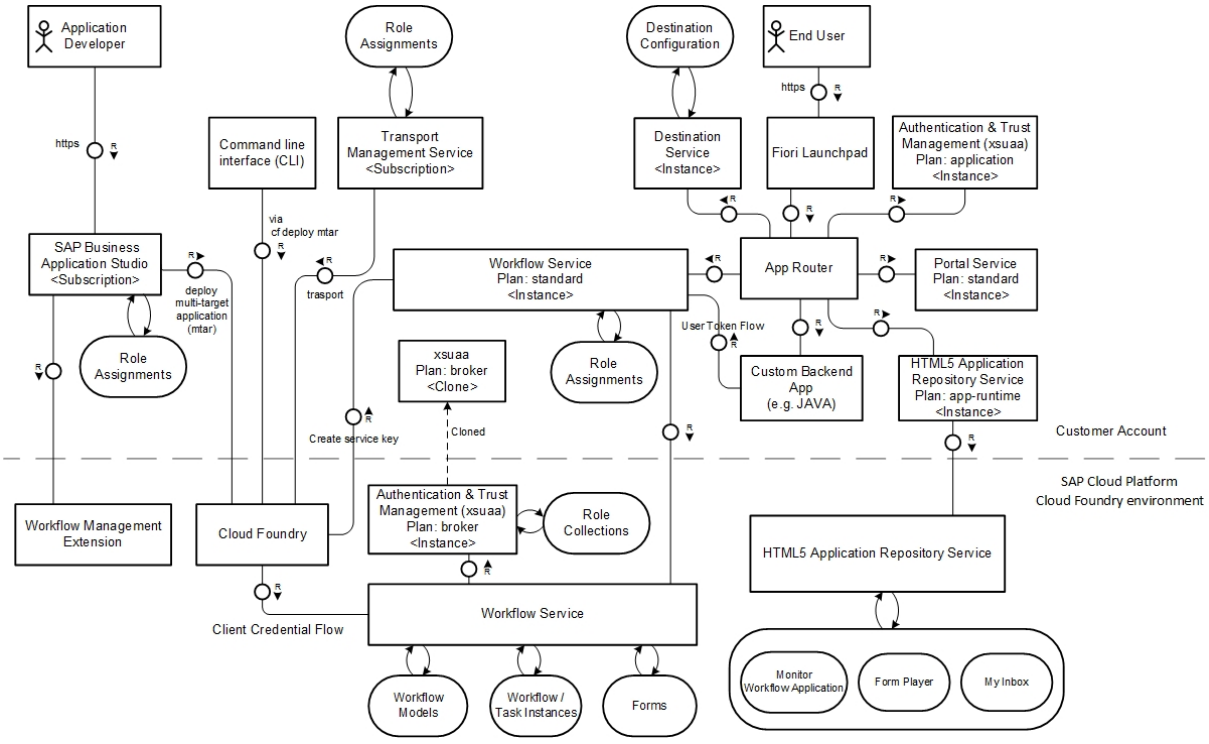
It does not replace the administration guide that is available for productive operation.

Related Information

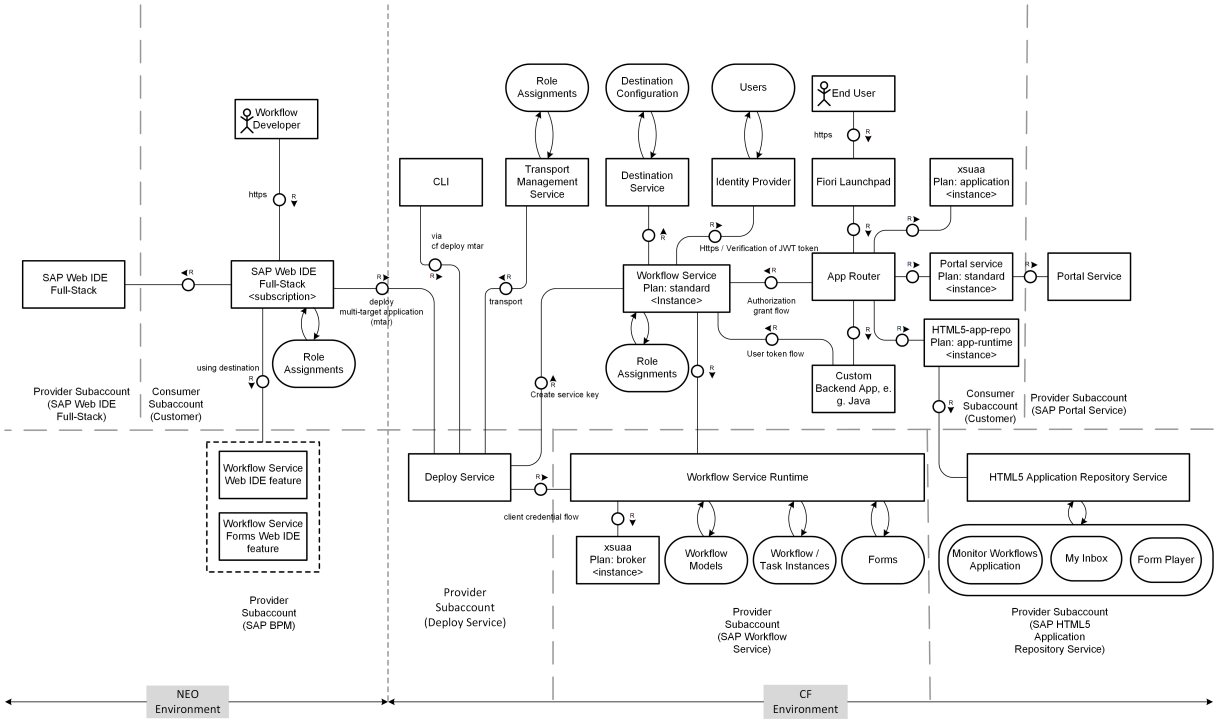
Security Guide for SAP Business Technology Platform (SAP BTP)

5.1 Architecture

The architecture of the workflow service comprises several components and subservices.



Overview of Components and Subservices When Using SAP Business Application Studio



Overview of Components and Subservices When Using SAP Web IDE Full-Stack

The workflow service includes the following subservices that are provisioned into the customer subaccount using the SAP Business Technology Platform (SAP BTP) cross-subaccount subscription concept:

- Workflow and form editors in SAP Business Application Studio or SAP Web IDE Full-Stack
- Workflow service runtime
- Monitor workflows
- My Inbox

For more information, see Multitenant Applications in the SAP Business Technology Platform (SAP BTP) documentation.

Prerequisites for using the workflow service:

- If you want to use My Inbox and the Monitor Workflows app, then a subscription to the SAP Cloud Portal service is required. See Create Workflow and My Inbox Tiles on Central Launchpad [page 184] and Configure My Inbox App [page 185].
- The workflow service runtime exposes a set of REST-based application programming interfaces (APIs) for managing workflow instances and task instances.
- Access to all subservices of the workflow service requires a valid user identity in the corresponding identity provider that is configured in the customer subaccount.
  For more information, see Identity Provider and Identity Management [page 229].

5.2 Identity Provider and Identity Management

For identity management and authentication, the workflow service relies on the identity provider (IdP) that is configured in the customer subaccount that owns the respective subscriptions.

All requests handled by the workflow service subscriptions are authenticated against the identity provider of the customer subaccount and authorized against the role assignments specified on the subscriptions in the customer subaccount. All users who need to interact with the various subservices of the workflow service must be available in the respective identity provider. You can replace the default Identity Authentication service with your own corporate identity provider.

For more information about the concepts and the necessary configuration steps, see Authorization and Trust Management in the Cloud Foundry Environment in the SAP Business Technology Platform (SAP BTP) documentation.

> **i Note**
>
> Changes to the identity provider can cause running, erroneous, and suspended workflow instances with principal propagation to fail on service tasks as soon as these are reached. It is extremely costly to recover such instances.

5.3 Authorization Configuration

Assign roles to specific users using the workflow service instance.

Assign authorities to OAuth clients using OAuth2 client credentials grant.

The workflow service runtime is one of many SAP Business Technology Platform (SAP BTP) services that you can subscribe to. A service instance of the workflow service is created when you subscribe to the workflow service.

Authorization for the workflow service is provided at the following levels:

- Workflow service global roles: Users who are assigned to these global roles are then granted the associated permissions for all workflow definitions, instances, and tasks.
- Workflow service authorities: OAuth clients are granted the associated permissions (based on a list of granted authorities) for all workflow definitions, instances, and tasks. For more information, see Technical Authentication [page 232].
- Instance-specific authorizations: Users who are assigned to these roles are granted permission only for the respective workflow instance. The workflow service provides APIs that use these authorizations. Users who are explicitly named by user ID, or as members of explicitly named groups, gain the associated permission only for the respective workflow instance. You can assign instance-specific permissions using the REST API. For more information, see Workflow Definition versus Workflow Instance [page 5].

Authorizations are cumulative: If any one authorization allows access, access is granted.

SAP Business Technology Platform (SAP BTP) includes predefined platform roles that support the typical tasks performed by users when interacting with the platform. In addition, subaccount administrators can combine various scopes into a custom platform role that addresses their individual requirements. Certain

activities, such as deployment of applications and assignment of roles to users or groups, require platform roles. These roles are assigned in the SAP BTP cockpit.

In addition, users must be assigned to the SAP Business Technology Platform (SAP BTP) `Space Developer` role.

For more information about assigning global roles and permissions, see Assign Workflow Roles to Your Users [page 19].

Roles for Accessing Workflow Service Runtime

Role	Description
`WorkflowDeveloper` (global role)	• Permission to query workflow definitions • Permission to retrieve the current error messages of a workflow instance • Permission to retrieve the model of the latest version of a specified workflow definition • Permissions to create and manage process templates

`WorkflowContextAdmin` (global role)
`contextAdminUsers`
`contextAdminGroups`
	• Permission to partially modify or completely override the workflow context of a workflow instance
	• Permission to retrieve the context of a task instance

`WorkflowContextViewer` (global role)
`contextViewerUsers`
`contextViewerGroups`
	• Permission to retrieve the context of a workflow instance
	• Permission to retrieve the context of a task instance

`WorkflowInitiator` (global role)	• Permission to view the sample context of a workflow definition • Permission to start workflow instances (using the API or the Monitor Workflows app)

`WorkflowParticipant` (global role)	• Permission to view tasks in My Inbox, where the user assigned to this role is a recipient
	• Permission to perform task operations including the following:
	  ○ Claim
	  ○ Release
	  ○ Call the task completion API
	• This role is a prerequisite to work with instance-specific permissions.

Role	Description

`WorkflowAdmin` (global role)
`adminUsers`
`adminGroups`
	• Permission to manage workflow definitions and workflow instances in the Monitor Workflows app*
	• Permission to query workflow definitions as well as query and cancel workflow instances*
	• Permission to retrieve and modify the tasks of a workflow instance
	• Permission to retrieve the current error messages of a workflow instance
	• Permission to retry the failed steps of an erroneous workflow instance
	• Permission to suspend and resume a workflow instance for temporary suspension of processing
	• Permission to retrieve the workflow logs for a given workflow instance
	• Permission to download the workflow model in the Monitor Workflow app*

`WorkflowMessageSender` (global role)	• Permission to send a message to a set of workflow instances for consumption in intermediate message events
`WorkflowTenantOperator` (global role)	**i Note** Consider carefully whether to assign the `WorkflowTenantOperator` role. Ideally, this should not be necessary in a productive system. • Permission to export data • Permission to undeploy workflow definitions • Permission to delete multiple workflow instances • Permission to purge all workflow definitions, workflow instances, and form definitions
`WorkflowEventSubscriber` (global role)	• Permission to subscribe to events. For internal use only.

`WorkflowViewer` (global role)
`viewerUsers`
`viewerGroups`
	• Permission to query workflow definitions* as well as query workflow instances
	• Permission to retrieve the tasks of a workflow instance
	• Permission to retrieve the workflow logs for a given workflow instance
	• Permission to download the workflow model

`WorkflowBusinessExpert` (global role)	• Permission to work with process variants.

* Only for global roles

Available Platform Roles for Using the Workflow Service Runtime

Role	Description
`Space Developer`	• Permission to deploy a workflow project

5.3.1 Technical Authentication

Access the workflow service with a technical user.

Technical scenarios might require a technical authentication without having a user authenticated through an identity provider. You can use the OAuth2 client credentials grant to authorize REST calls for your tenant.

Each service instance of the workflow service provides an OAuth2 client through the service binding. When creating the service instance, you can specify a list of authorities to get granted to the related OAuth2 client during client credentials grant.

Technical authentication is not possible for operations that are configured to propagate the user to a subsequent service task. Principal propagation requires a user authenticated through an identity provider.

The technical authentication has no relation to a user authenticated by an identity provider. Storing and displaying the human user is not possible when using technical authentication even though a human user implicitly triggered the action. Instead the technical name of the OAuth client is used. For example, technical logs and audit logs store the OAuth client ID if technical authentication is used.

> → Tip
>
> Everybody who has access to the OAuth2 client can execute actions anonymously based on the authorizations granted to the OAuth2 client. Limit the number of authorizations granted to a single service instance. You can create dedicated service instances for:
>
> - Different usage scenarios that have the minimum authorizations required for the respective scenario
> - Technical authentication and for regular user authentication

Related Information

Enable Technical Authentication [page 174]

5.4 Destinations

Subservices communicate using predefined destinations in a customer subaccount, for example, when the My Inbox or the *Manage Workflows* application communicates with the workflow runtime.

Predefined destinations are generated and configured when you enable the workflow service in a customer subaccount. For more information, see Principal Propagation for User Interfaces [page 233] below.

The workflow runtime communicates, according to the workflow definitions, with other services.

- The workflow runtime uses destinations of type `Mail` to communicate with mail servers.
  For more information, see Configure the Workflow Service Mail Destination [page 177] and Configure Mail Tasks [page 60].
- To communicate with other services, the workflow runtime uses destinations of type `HTTP`.

For more information, see Destination Configuration for Service Task [page 233] below and Configure Service Tasks [page 45]. For authentication and authorization purposes, also other, referenced destinations might be used, for example, OAuth2 authorization endpoints.

Secure Communication Protocols and Cipher SuitesSecure Communication Protocols and Cipher Suites

We recommend that the services or servers maintained in the destinations support at least the secure communication protocol TLS 1.2. The list of supported cipher suites might vary between private and public cloud deployments and/or data centers.

Note, the supported protocol versions as well as cipher suites are subject to change and deprecation for improved security.

Principal Propagation for User Interfaces

Communication between different subservices uses principal propagation, which forwards the user who is logged on to the user interface to the workflow service runtime. This lets all requests that are sent to the workflow service runtime on behalf of the user (who initiated the request from the user interface) be posted.

Principal propagation is automatically enabled when you enable the workflow service in a customer subaccount.

Destination Configuration for Service Tasks

The workflow service supports outbound connectivity for orchestrating external services and systems. Destinations decouple modeling service tasks in your workflow model from the configuration of the physical back-end systems that are called in the service task at runtime.

> → Tip
>
> Configure destinations to use secure communication protocols, such as HTTPS, wherever possible.

While the standard destination concept in SAP Business Technology Platform (SAP BTP) can be used for this purpose, there are several limitations that apply to their usage in the context of the workflow service.

You can call services using the following authentication types:

- Basic Authentication: Select *Basic Authentication* as the authentication type in the destination.

  > i Note
  >
  > For On-Premise connections, use *SAP Connectivity service* (Cloud Connector) Version 2.11 or higher. Ensure that the *Principal Type* of the respective *Cloud To On-Premise connection* is set to *X.509 certificate* (strict usage). For more information, see Configure Access Control (HTTP) in the SAP Connectivity service documentation.

- OAuth2 Client Credentials flow: Select *OAuth2ClientCredentials* as the authentication type in the destination, see OAuth Client Credentials Authentication.
- OAuth2SAMLBearerAssertion: For calls to services outside of SAP Cloud Foundry. Use this authentication type to propagate the user from certain actions on the workflow to other services. For more information, see Configure a Service Task Destination with OAuth2SAMLBearerAssertion for Principal Propagation [page 235] and Configuring Principal Propagation for Service Tasks [page 176].
- OAuth2UserTokenExchange: For calls to services in SAP Cloud Foundry. Use this authentication type to propagate the user from certain actions on the workflow to other services. For more information, see OAuth User Token Exchange Authentication and Configuring Principal Propagation for Service Tasks [page 176].
- No Authentication: If the service you want to call doesn't require any authentication, select *No Authentication* as the authentication type in the destination.

Besides the authentication type, the following destination features are supported in the workflow service:

- Server authentication (verification): JDK default and custom truststores
- Supported proxy type: *Internet*, *OnPremise*
- Destination type:
  - Supports HTTP and HTTPS connectivity based on HTTP destinations in SAP Business Technology Platform (SAP BTP).
  - For an *OnPremise* destination, you can optionally specify the `LocationId` property.

To connect to on-premise back-end systems, you can use the SAP Connectivity service. For more information about how to install and configure the SAP Connectivity service, see the SAP Connectivity service documentation.

To configure destinations, use the standard SAP Business Technology Platform (SAP BTP) mechanisms in the SAP BTP cockpit. For more information, see Managing Destinations.

> **i Note**
>
> For server verification, additional properties that were configured at the destinations as described in Server Certificate Authentication are ignored. Consequently, you can't turn off trust verification, and host names are always verified in strict mode.
>
> Workflow service does not support certificate revocation checks, such as using Certificate Revocation Lists or Online Certificate Status Protocol.
>
> If you use the `OnPremise` proxy type to connect to an on-premise back-end system, make sure that you specify the URL of the virtual host that is maintained in the SAP Connectivity service as the destination URL, rather than the actual URL of the back-end system. The scheme of the specified URL must be `http://`, not `https://`.

While destination configuration data is stored completely within the customer subaccount, the workflow service runtime must temporarily access this data when executing a workflow instance. This data isn't persisted within the workflow service itself.

5.4.1 Configure a Service Task Destination with OAuth2SAMLBearerAssertion for Principal Propagation

You can set up destinations that use OAuth2SAMLBearerAssertion for Principal Propagation.

Prerequisites

- Model a service task in a workflow model.
- Create a destination in your account where the name matches the destination property in the service task. In addition, make sure the URL points to the service you want to call.
  For more information, see Managing Destinations.
- Verify that the service, which the service task should call, is protected with OAuth2 and supports the OAuth2 SAML bearer assertion flow.
- Know the corresponding OAuth2 token endpoint URL, client ID, and client secret.

Context

> **i Note**
>
> This destination authentication type doesn't work for calls from the SAP BTP, Cloud Foundry environment to the SAP BTP, Cloud Foundry environment. To find the correct destination authentication type for this purpose, see Destinations [page 232].

Procedure

1. Edit or create a destination as described in Managing Destinations.
2. Set *OAuth2SAMLBearerAssertion* as the authentication type.
3. Configure the destination using the following data. Also maintain the additional properties necessary for your particular service provider. For more information, see SAML Bearer Assertion Authentication.

Property	Value
`audience`	`entityID` property of the SAML 2.0 metadata. For SAP BTP, Neo environment, see Principal Propagation to OAuth-Protected Applications For SAP BTP, Cloud Foundry environment, see Principal Propagation from the Neo to the Cloud Foundry Environment. To access a SAP BTP, Neo environment service from the SAP BTP, Cloud Foundry environment, you need to configure trust between the SAP BTP, Cloud Foundry environment and SAP BTP, Neo environment. See, for example, Principal Propagation from the Cloud Foundry to the Neo Environment. For more information, see the documentation of your service provider.
`URL`	Endpoint of the service you want to call
`clientKey`	Client ID of the OAuth client
`tokenServiceURL`	Token endpoint URL of the OAuth server Example of what the URL could look like: In the SAP BTP, Neo environment: **`https://oauthasservices-<subaccount>.<region>/oauth2/api/v1/token`**
`tokenServiceUser`	Client ID of the OAuth client
`tokenServicePassword`	Client secret of the OAuth client

If you want to call a service in the SAP BTP, Cloud Foundry environment, see the client ID in the cockpit in your space on the *Instances and Subscriptions* page. Select the row, then expand the details using the expand icon at the end of the row. There, you can view the *Service Key*.

If you want to call a service in the SAP BTP, Neo environment, see the client ID in the cockpit under *Security* > *OAuth* > *Clients*.

4. Add a new property:

Property	Value
`authnContextClassRef`	`urn:oasis:names:tc:SAML:2.0:ac:classes:X509`

5. (Optional) If your destination points to a service in a different subaccount in the SAP BTP, Neo environment or SAP BTP, Cloud Foundry environment, you must configure trust between these accounts.

   For more information, see Principal Propagation

> **i Note**
>
> Access permissions for a user involved in principal propagation are cached upon the latest interaction of this user with the workflow system. If you are using a custom identity provider for the workflow service and if the permissions of the user change, including the deletion of the user, those changes

might not be reflected upon the request to the target system of the service task. In this case, you might have to explicitly cancel and if needed also restart the workflow instance.

5.5 Data Protection and Data Privacy

Governments place legal requirements on industry to protect data and privacy. We provide features and functions to help you meet these requirements.

SAP does not provide legal advice in any form. SAP software supports data protection compliance by providing security features and data protection-relevant functions, such as blocking and deletion of personal data. In many cases, compliance with applicable data protection and privacy laws is not covered by a product feature. Furthermore, this information should not be taken as advice or a recommendation regarding additional features that would be required in specific IT environments. Decisions related to data protection must be made on a case-by-case basis, taking into consideration the given system landscape and the applicable legal requirements. Definitions and other terms used in this documentation are not taken from a specific legal source.

⚠ Caution

SAP Workflow service does not provide the technical capabilities to support the collection, processing, and storage of sensitive personal data..

→ Recommendation

Working copies of data from systems of record that are stored in a workflow context should be limited to the very minimum required for the processing.

5.5.1 Information Report

An information report is a collection of data relating to a data subject. A data privacy specialist may be required to provide such a report or an application may offer a self-service.

REST API endpoints help the data privacy specialist when building a report. The data export endpoint, for example, enables the data privacy specialist to retrieve all relevant information for further processing.

For more information, see Using Workflow APIs [page 151] and Export SAP Workflow Service Data [page 213].

5.5.2 Erasure

When handling personal data, consider local legislation. After the data has passed the end of purpose, regulations may require you to delete the data. However, additional regulations may require you to keep the data longer. During this period, you must block access to the data by unauthorized persons until the end of the retention period, when the data is finally deleted.

Personal data can also include referenced data. The challenge for deletion and blocking is first to handle referenced data and then other data, such as business partner data.

As part of the SAP Business Technology Platform (SAP BTP) offboarding process, all data stored within the workflow service is deleted.

For audit needs, the workflow service offers an export feature. For more information, see Export SAP Workflow Service Data [page 213].

To delete workflow data, the following APIs offer a broad scope of data deletion options:

- POST "`/v1/purge`" completely deletes all workflow data.
- DELETE "`/v1/workflow-definitions/{definitionId}`" deletes a single workflow definition and all its workflow instances and the related data.
- PATCH "`/v1/workflow-instances`" deletes a single workflow instance and all related data.
- PATCH "`/v1/workflow-instances/{workflowInstanceId}/context`" allows updating the workflow context to remove sensitive data, leaving the workflow instance in place.
- DELETE "`/v1/forms/{formId}`" deletes a single workflow form.

For more information, see the SAP Workflow service API.

> ⚠ Caution
>
> Workflow definitions and form definitions are persisted separately. Deleting a workflow definition does not delete dependent form definitions and the other way round.
>
> Deleting dependent artifacts of a workflow, such as form definitions, may break existing workflow definitions and running workflow instances.

5.5.3 Change Log

For auditing purposes or for legal requirements, changes made to personal data should be logged, enabling the monitoring of who made changes and when.

Therefore, SAP Workflow service may write logs into the audit log handled by the platform itself.

> i Note
>
> SAP Workflow service does not provide inherent support for logging changes in the workflow context.
>
> The workflow developer must take care of logging changes to attributes in the workflow context that hold personal data. Such changes may occur, for example, when calling external services, through intermediate message events, or when updating context data through the REST API.

Workflow definitions may include personal data, for example, the user IDs of task recipients. For this kind of data, the API provides versioning access at `/v1/workflow-definitions/{definitionId}/versions`.

The workflow service contains information about which users completed which tasks. You can retrieve this information using the REST API endpoint `/v1/workflow-instances/{workflowInstanceId}/execution-logs`.

Furthermore, it contains information about which user has deployed a form definition. You can retrieve this information by using the data export endpoint, see Information Report [page 237].

5.5.4 Glossary

Term	Definition
Blocking	A method of restricting access to data for which the primary business purpose has ended.
Business purpose	A legal, contractual, or in other form justified reason for the processing of personal data. The assumption is that any purpose has an end that is usually already defined when the purpose starts.
Consent	The action of the data subject confirming that the usage of his or her personal data shall be allowed for a given purpose. A consent functionality allows the storage of a consent record in relation to a specific purpose and shows if a data subject has granted, withdrawn, or denied consent.
Deletion	Deletion of **personal data** so that the data is no longer available.
End of business	Date where the business with a data subject ends, for example the order is completed, the subscription is canceled, or the last bill is settled.
End of purpose (EoP)	End of purpose and start of blocking period. The point in time, when the primary processing purpose ends (for example contract is fulfilled).
End of purpose (EoP) check	A method of identifying the point in time for a data set when the processing of **personal data** is no longer required for the primary **business purpose**. After the **EoP** has been reached, the data is **blocked** and can only be accessed by users with special authorization (for example, tax auditors).
Personal data	Any information relating to an identified or identifiable natural person ("data subject"). An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural, or social identity of that natural person
Purpose	The information that specifies the reason and the goal for the processing of a specific set of personal data. As a rule, the purpose references the relevant legal basis for the processing of personal data.

Term	Definition
Residence period	The period of time between the end of business and the end of purpose (EoP) for a data set during which the data remains in the database and can be used in case of subsequent processes related to the original purpose. At the end of the longest configured residence period, the data is blocked or deleted. The residence period is part of the overall retention period.
Retention period	The period of time between the end of the last business activity involving a specific object (for example, a business partner) and the deletion of the corresponding data, subject to applicable laws. The retention period is a combination of the residence period and the blocking period.
Sensitive personal data	A category of personal data that usually includes the following type of information: • Special categories of personal data, such as data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, trade union membership, genetic data, biometric data, data concerning health or sex life or sexual orientation. • Personal data subject to professional secrecy • Personal data relating to criminal or administrative offenses • Personal data concerning insurances and bank or credit card accounts
Where-used check (WUC)	A process designed to ensure data integrity in the case of potential blocking of business partner data. An application's where-used check (WUC) determines if there is any dependent data for a certain business partner in the database. If dependent data exists, this means the data is still required for business activities. Therefore, the blocking of business partners referenced in the data is prevented.

5.6 Audit Logs

The workflow service writes entries into the audit log of the consumer account for the following operations:

For information on audit logging and on how to access the logs, see *Related Information*.

Tenant Operations

- Create new tenant
  Audit log message category: `SecurityEventAuditMessage`
- Delete existing tenant
  Audit log message category: `SecurityEventAuditMessage`
- Purge all workflow data of a tenant
  Audit log message category: `SecurityEventAuditMessage`
- Export all workflow data from a tenant
  Audit log message category: `SecurityEventAuditMessage`

Service Instances and Subscriptions

- Create a new service instance
  Audit log message category: `SecurityEventAuditMessage`
- Delete an existing service instance
  Audit log message category: `SecurityEventAuditMessage`
- Update an existing service instance
  Audit log message category: `SecurityEventAuditMessage`
- Create a new subscription
  Audit log message category: `SecurityEventAuditMessage`
- Delete an existing subscription
  Audit log message category: `SecurityEventAuditMessage`

Deployment

- Deploy a workflow definition
  Audit log message category: `SecurityEventAuditMessage`
- Undeploy a workflow definition
  Audit log message category: `SecurityEventAuditMessage`
- Deploy a form definition
  Audit log message category: `SecurityEventAuditMessage`
- Undeploy a form definition
  Audit log message category: `SecurityEventAuditMessage`

Workflow Instances

- Delete a workflow instance
  Audit log message category: `SecurityEventAuditMessage`

Event Subscriptions

- Create, update, and delete event subscriptions
  Audit log message category: `ConfigurationChangeAuditMessage`

Security

- Update instance-specific role assignments of a workflow instance
  Audit log message category: `ConfigurationChangeAuditMessage`

Substitution

- Create, update, and delete event substitution rules
  Audit log message category: `SecurityEventAuditMessage`

Related Information

Audit Logging in the Cloud Foundry Environment

6 Monitoring and Troubleshooting

When working with the workflow service, you may encounter issues that prevent access or affect performance.

Getting Support

If you encounter an issue with this service, we recommend to follow the procedure below:

Check Platform Status

Check the availability of the platform at SAP Trust Center.

For more information about selected platform incidents, see Root Cause Analyses.

Check Guided Answers

In the SAP Support Portal, check the Guided Answers section for SAP Business Technology Platform (SAP BTP) as well as the Guided Answers for Workflow Service.

Contact SAP Support

You can report an incident or error through the SAP Support Portal. Please use the following component for your incident:

Component Name	Component Description
LOD-BPM-WFS	Workflow Service Runtime / Editor

When submitting the incident we recommend to include the following information:

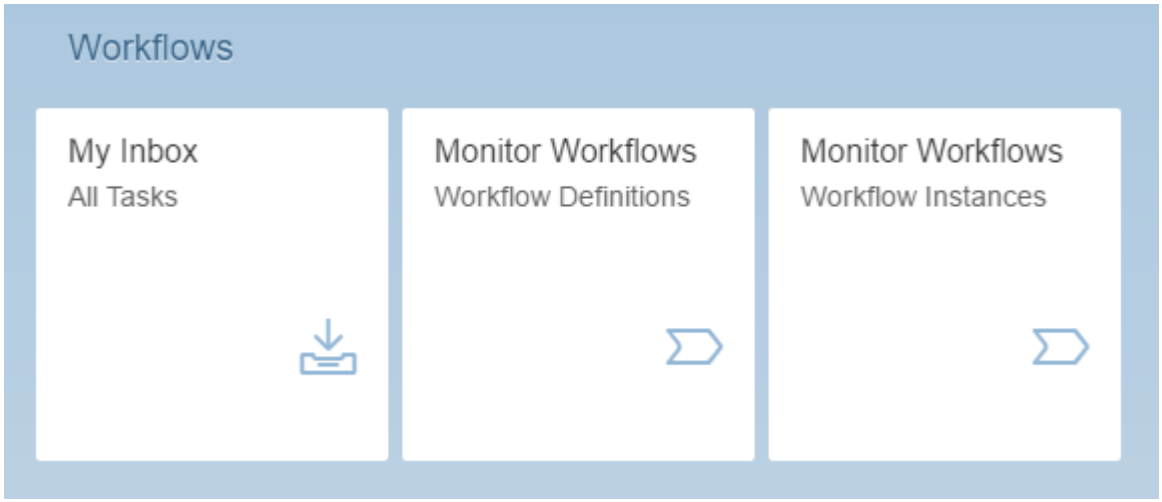
- Landscape information (Canary, EU10, US10)
- The URL of the page where the incident or error occurs
- The steps or clicks used to replicate the error
- Screenshots, videos, or the code entered

6.1 Managing Workflows Using the Monitor Workflows App

With the web-based administration Monitor Workflows app you can manage workflow instances and workflow definitions.

The app offers two interlinked views, one for workflow instances and one for workflow definitions. Both can be accessed using dedicated tiles in the SAP Fiori launchpad.

For information on how to access these tiles, see Create Workflow and My Inbox Tiles on Central Launchpad [page 184].



> **i** Note
>
> You must have the latest maintenance version or latest version of SAP UI5 configured on SAP Fiori launchpad to use the Monitor Workflows app.

Related Information

Managing Workflow Instances [page 244]
Managing Task Instances [page 246]
Managing Workflow Definitions [page 248]
Deep Linking in Monitor Workflows App [page 250]

6.1.1 Managing Workflow Instances

The workflow instances view shows a list of all workflow instances and offers actions to work on the instances.

Prerequisites

The SAP Fiori launchpad objects are configured. For more information, see Create Workflow and My Inbox Tiles on Central Launchpad [page 184].

Available Actions

Actions in the list of workflow instances:

- Search for workflow instances using the following criteria: workflow ID, workflow definition ID, subject, business key, or the initiator of the workflow instance.
  Type the keyword you want to use in the *Search* field, and choose *(Search)*, or choose Enter.
- Filter for workflow instances based on their status and definition. You can also filter only the root workflow instances or all workflow instances that include subflows.
  Choose *(Filter)*.

  > **i Note**
  >
  > - By default, the filter is applied to show workflow instances in running, erroneous, and suspended status. However, you can also filter for workflow instances in status *Completed* and *Canceled*.
  > - By default, the filter is applied to show root workflow instances. You can choose to filter all the workflow instances including the subflow instances by choosing *All Instances*.

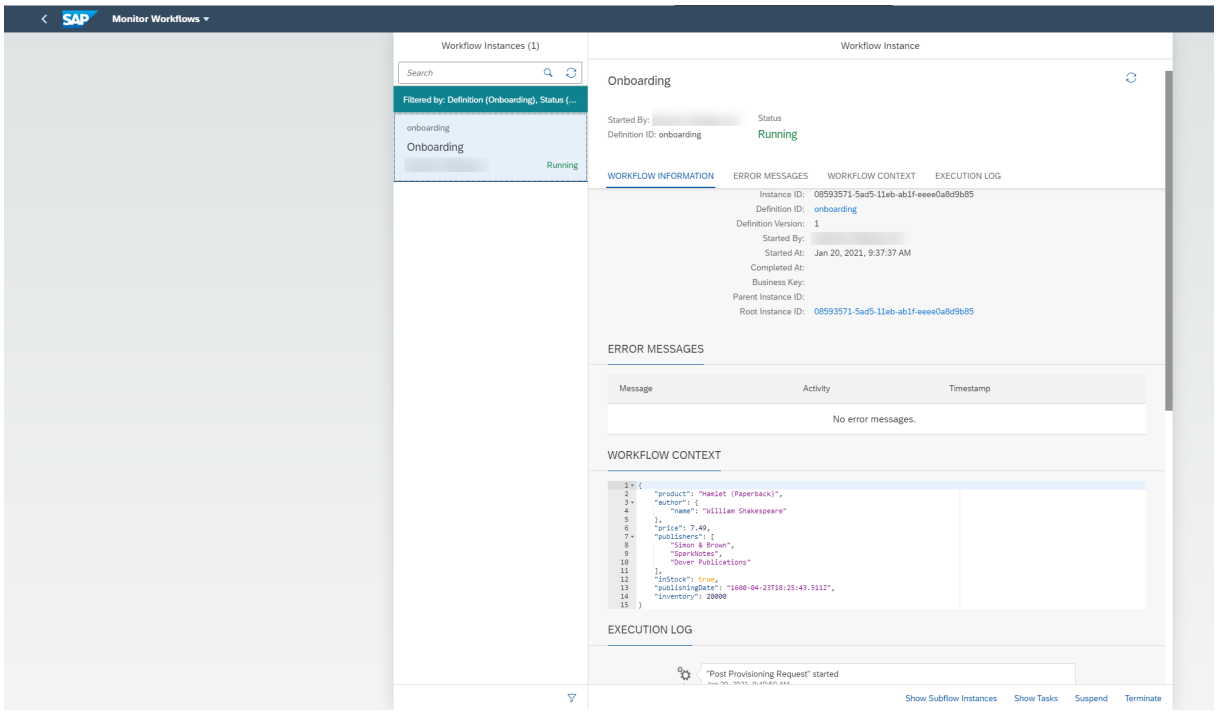
- Display details about a workflow instance and navigate to it by selecting a workflow instance.

Actions in the details screen of the workflow instance:

- View details of a workflow on the *Workflow Information* tab.
- View the current context of a workflow instance on the *Workflow Context* tab.
- View the executed activities in the workflow instance and their main details on the *Workflow Execution Log* tab. For more information, see Workflow Execution Log [page 166].
- View the subflow instances that are started by a workflow instance. Choose *Show Subflow Instances*.
- View tasks of a workflow instance. Choose *Show Tasks*.
- Retry the execution of failed steps of an erroneous workflow instance. Choose *Retry*.
- Cancel a running workflow instance. Choose *Terminate*.
- Load more entries. Scroll down to the end of the list and choose *More*.
- View the count of instances displayed in the view versus the total instance count.
- Suspend a running or erroneous workflow instance. Choose *Suspend*.
- Resume a suspended workflow instance. Choose *Resume*. This operation also retries failed steps.
- Navigate to the workflow definition of an instance. Click the workflow definition ID.
- Navigate to the parent workflow instance for a subflow. Click the *Parent Instance ID* link.
- Navigate to the root instance of a workflow or subflow. Click the *Root Instance ID* link.

  > **i Note**
  >
  > Root instances are main workflow instances that are started by an application or a user. Main workflow instances in turn start subflow instances.

Screenshot of the Workflow Instances



Related Information

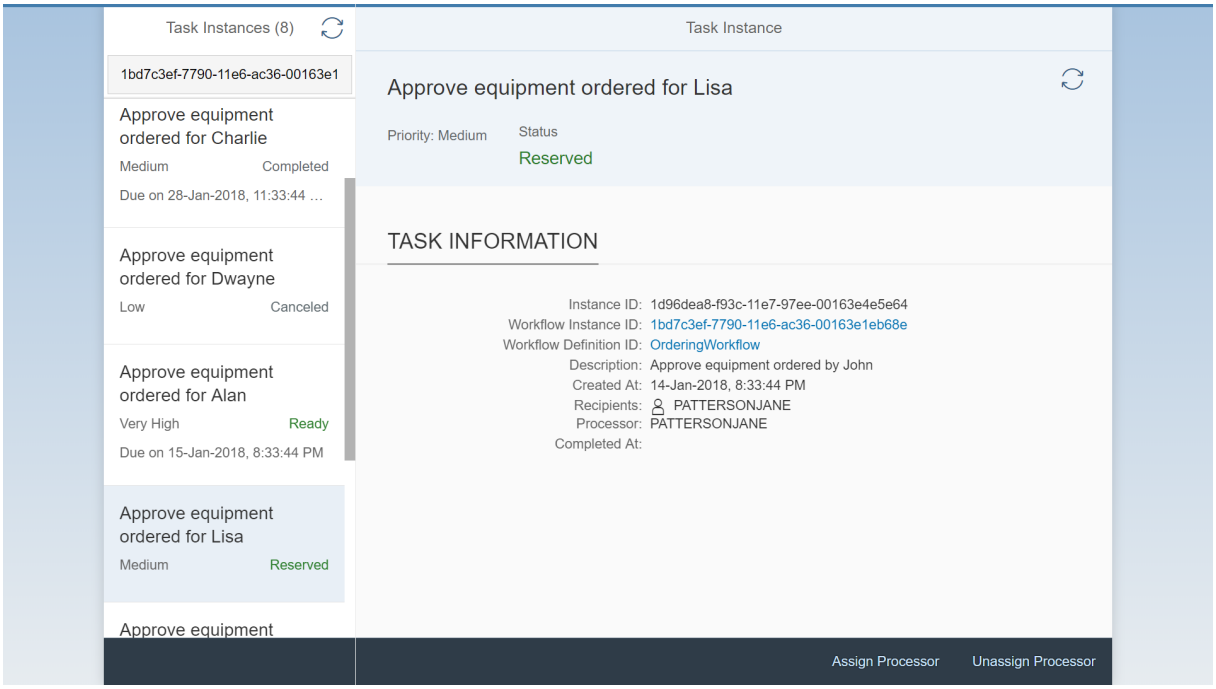
Managing Workflows Using the Monitor Workflows App [page 243]
Deep Linking in Monitor Workflows App [page 250]
Blog: Controlling User Access to Monitor Workflows in SAP BTP

6.1.2 Managing Task Instances

The task instances view shows tasks for a given workflow instance.

Prerequisites

The SAP Fiori launchpad objects are configured. For more information, see Create Workflow and My Inbox Tiles on Central Launchpad [page 184].



The following actions are available:

- To display details about a task instance, select the task.
- To navigate to the workflow instance of the task, click the workflow instance ID on the details screen.
- To navigate to the workflow definition, click the workflow definition ID on the details screen.
- To assign a processor for a task, choose *Assign Processor*.
  This action is only available for tasks with status Ready or Reserved.
  When you assign a task to a user, the user becomes its processor. All other task recipients can no longer see the task in My Inbox. The status of the task changes from Ready to Reserved.
- To unassign the processor of a task, choose *Unassign Processor*.
  This action is only available for tasks with status Reserved.
  When you unassign a task from a processor, it is available again to all recipients and they can see the task in My Inbox. The status of the task changes from Reserved to Ready.

Related Information

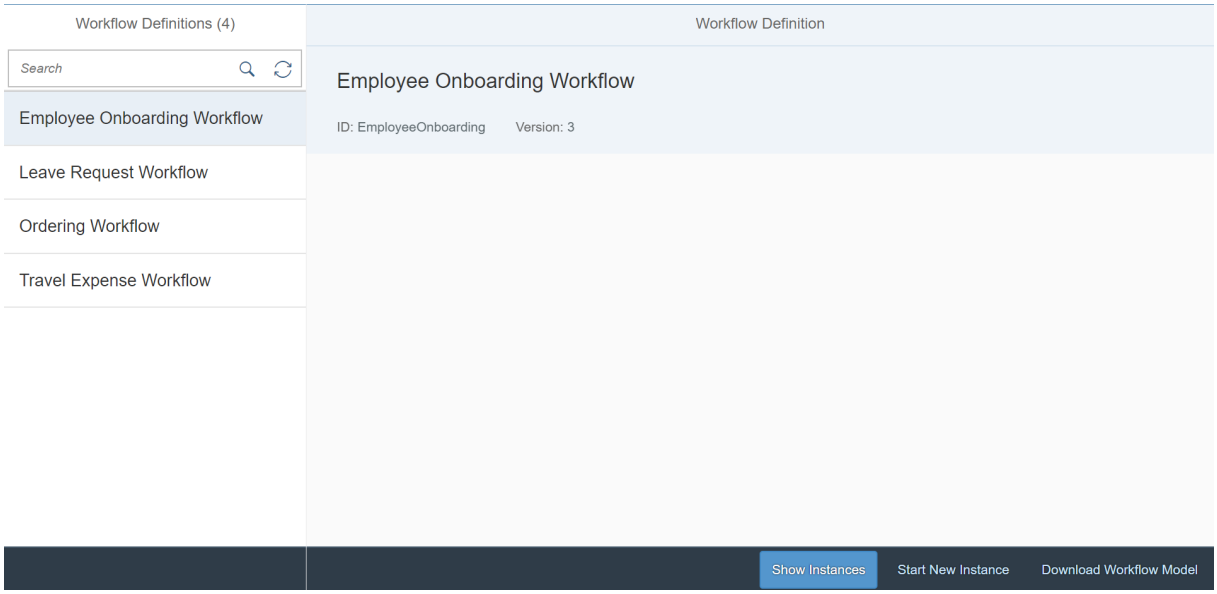
Managing Workflows Using the Monitor Workflows App [page 243]
Deep Linking in Monitor Workflows App [page 250]

6.1.3 Managing Workflow Definitions

The workflow definitions view shows a list of deployed workflow definitions.

Prerequisites

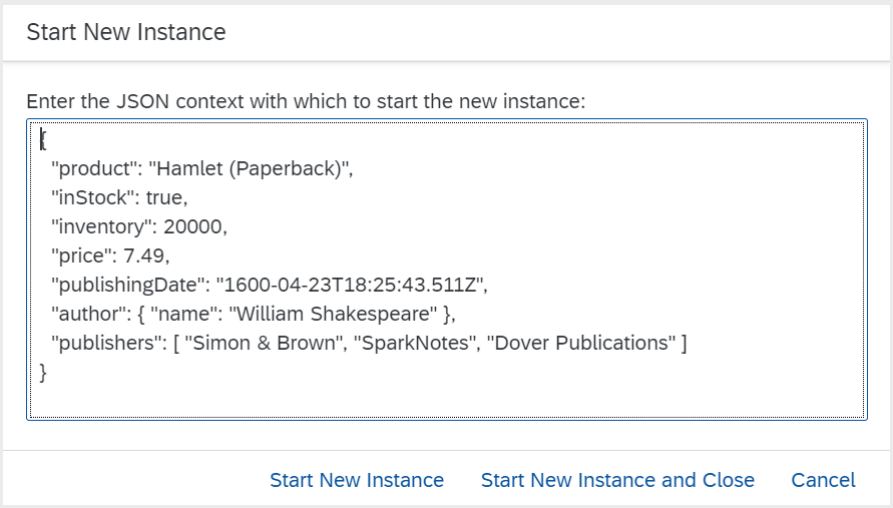
The SAP Fiori launchpad objects are configured. For more information, see Create Workflow and My Inbox Tiles on Central Launchpad [page 184].



The following actions are available:

- To filter the workflow definitions, use the following criteria: workflow definition ID, workflow definition name, or the workflow definition version.
- To search the workflow definitions, type the keyword you want to use in the *Search* field, and choose 🔍 *(Search)*, or press *Enter*.
- To start a new workflow instance, select a workflow definition and choose *Start New Instance*. You can also choose *Start New Instance and Close*, where the dialog closes upon starting an instance.
  If you've configured a sample context while modeling a start event, it's shown as the context data while starting a new workflow instance in the *Start New Instance* window. However, you can also modify this JSON context data as required. For more information, see Configure Start Events [page 65].

Example



The JSON structure contains the content to be passed to the workflow context. In contrast to the workflow service API, a context node as a wrapper isn't required.

> **Note**
>
> In the workflow context, use numbers where computations or comparisons on them are required. We recommend that you don't use numbers as IDs, especially not for business keys. Use a string instead.

For more information about using these actions, see the Workflow Service API documentation in Using Workflow APIs [page 151].

- To navigate to the list of all instances of a definition, select the definition from the list and choose *Show Instances*.
- To load more workflow definitions, scroll down to the end of the list and choose *More*.
- To download the workflow model, select the definition from the list, then choose *Download Workflow Model*. With this, you retrieve the workflow model for the latest deployed version of a workflow definition.

> **Note**
>
> We recommend that you don't import this downloaded workflow model to SAP Web IDE Full-Stack.

Related Information

Managing Workflows Using the Monitor Workflows App [page 243]
Deep Linking in Monitor Workflows App [page 250]

6.1.4 Deep Linking in Monitor Workflows App

You can access the workflow definitions, instances, and task instances using direct URLs. You can use the below URL formats to access the required information.

Workflow Instance

- To access the list of workflow instances, use the following URL format:
  `https://flpsandbox-<consumer_account>.<landscape_host>/sites?siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances`
- To access a particular workflow instance, use the following URL format:
  `https://flpsandbox-<consumer_account>.<landscape_host>/sites?siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances/<workflow_instance_id>`

Workflow Definition

- To access the list of workflow definitions, use the following URL format:
  `https://flpsandbox-<consumer_account>.<landscape_host>/sites?siteId=<site_id>#bpmworkflowmonitor-DisplayDefinitions&/workflowDefinitions`
- To access a particular workflow definition, use the following URL format:
  `https://flpsandbox-<consumer_account>.<landscape_host>/sites?siteId=<site_id>#bpmworkflowmonitor-DisplayDefinitions&/workflowDefinitions/<workflow_definition_id>`

Task Instance

- To access the list of task instances for a particular workflow instance, use the following URL format:
  `https://flpsandbox-<consumer_account>.<landscape_host>/sites?siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances/<workflow_instance_id>/taskInstances`
- To access a particular task instance, use the following URL format:
  `https://flpsandbox-<consumer_account>.<landscape_host>/sites?siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances/<workflow_instance_id>/taskInstances/<task_instance_id>`

> **i Note**
>
> If you are using the default site, then `site_id` in the URL is not mandatory.

Related Information

Managing Workflow Instances [page 244]
Managing Task Instances [page 246]
Managing Workflow Definitions [page 248]

6.2 Traceability during Transport of Workflow Modules

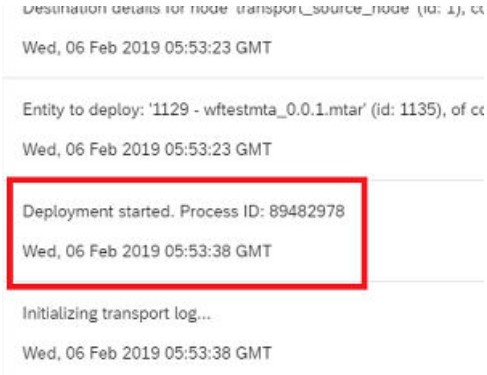
You can use the transport and deploy logs to trace the details while transporting an MTA archive between two subaccounts.

Prerequisites

- You have the multitarget application archives (MTAs -.mtar files) transported between two subaccounts. For more information, seeWhat Is Cloud Transport Management.
- You have downloaded and installed the Cloud Foundry Command Line Interface. For more information, see Download and Install the Cloud Foundry Command Line Interface.
- You have installed the multitarget application plug-in in the Cloud Foundry environment. For more information, see Install the Multitarget Application Plug-in in the Cloud Foundry Environment.

Procedure

1. Get the *Process ID* using the following steps:
    1. In the **Transport Action Logs**, select the required source node with the *Action Type* as **Import to Node** for a specific user.
    2. Note the *Process ID* from the log.



2. Log on to the Cloud Foundry environment using the Cloud Foundry Command Line Interface. For more information, see Log On to the Cloud Foundry Environment Using the Cloud Foundry Command Line Interface.

3. When prompted, select an *org*.
4. When prompted, select a *space*.
5. Download the deploy logs for the required *Process ID* by providing the command:

   ```
   cf dmol -i < Process ID>
   ```

6. Extract the downloaded `.zip` file and open the *MAIN_LOG* file.

> **i** Note
>
> In the *MAIN_LOG* file,
> - Using the value of *Authenticated user ID*, you can trace which user has transported the MTA archive.
> - Using the value of *deployResourceId*, you can see the transported workflows and forms with their ID, name, version, and status.

6.3 Verify the Availability of Workflow Extensions in SAP Business Application Studio

You've opened a Dev Space in SAP Business Application Studio and quickly want to check whether the workflow extension is loaded correctly.

Procedure

1. To open the *Plugins* view, choose *Menu* > *View* > *Plugins*.

   You should see the following entries:
   - workflow-editor
   - workflow-forms-editor

2. To verify whether the generators are loaded correctly, choose *Menu* > *Terminal* > *New Terminal*.

   To start the generator framework, type in the new terminal that opens `yo` and then choose ENTER.

   Verify that the following generator is part of the list: `@workflow/workflow Module`. If the entry appears, you're good to go.

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.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

www.sap.com/contactsap

THE BEST RUN SAP