PUBLIC

Document Version: 1.0 – 2025-02-27

Workflow in the Cloud Foundry Environment

THE BEST RUN

Content

1 Introduction

The workflow capability offers modern process automation capabilities.

You design workflows based on Business Process Model and Notation (BPMN) in a graphical editor. Within that editor, you can use forms to model workflow user interfaces. Users can then process their tasks contained in the workflow, using an inbox. You monitor and manage workflow definitions and workflow instances and can, for example, start workflow instances to enable smooth operation.

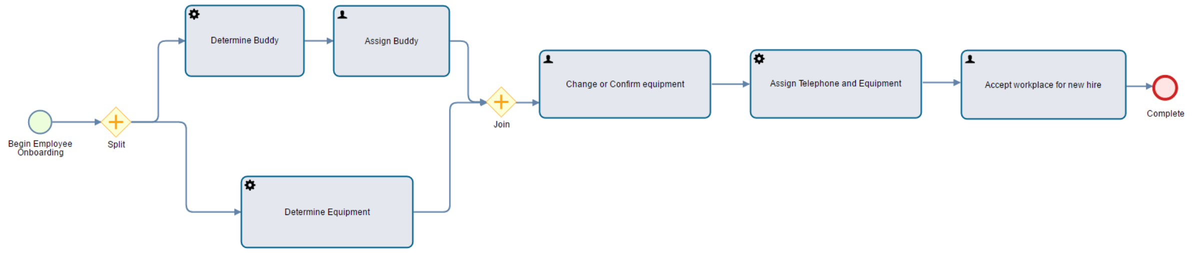
The following sections introduce some basic definitions and concepts. You also find information on conventions, restrictions, and limitations to help you to achieve optimal use of the service. Finally, there's general information on supported languages and browser support.

Related Information

Business Process Model and Notation
Workflow Definition versus Workflow Instance [page 5]
Status Changes for Workflow Instances [page 6]
Status Changes for Task Instances [page 7]
Conventions, Restrictions, and Limits [page 8]
Supported Languages [page 14]
Browser Support [page 16]

1.1 Workflow Definition versus Workflow Instance

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


Begin Employee Onboarding
Split
Determine Buddy
Assign Buddy
Determine Equipment
Join
Change or Confirm equipment
Assign Telephone and Equipment
Accept workplace for new hire
Complete

Example Workflow Depicted as a Diagram

The workflow 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. 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.

> ⓘ 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.

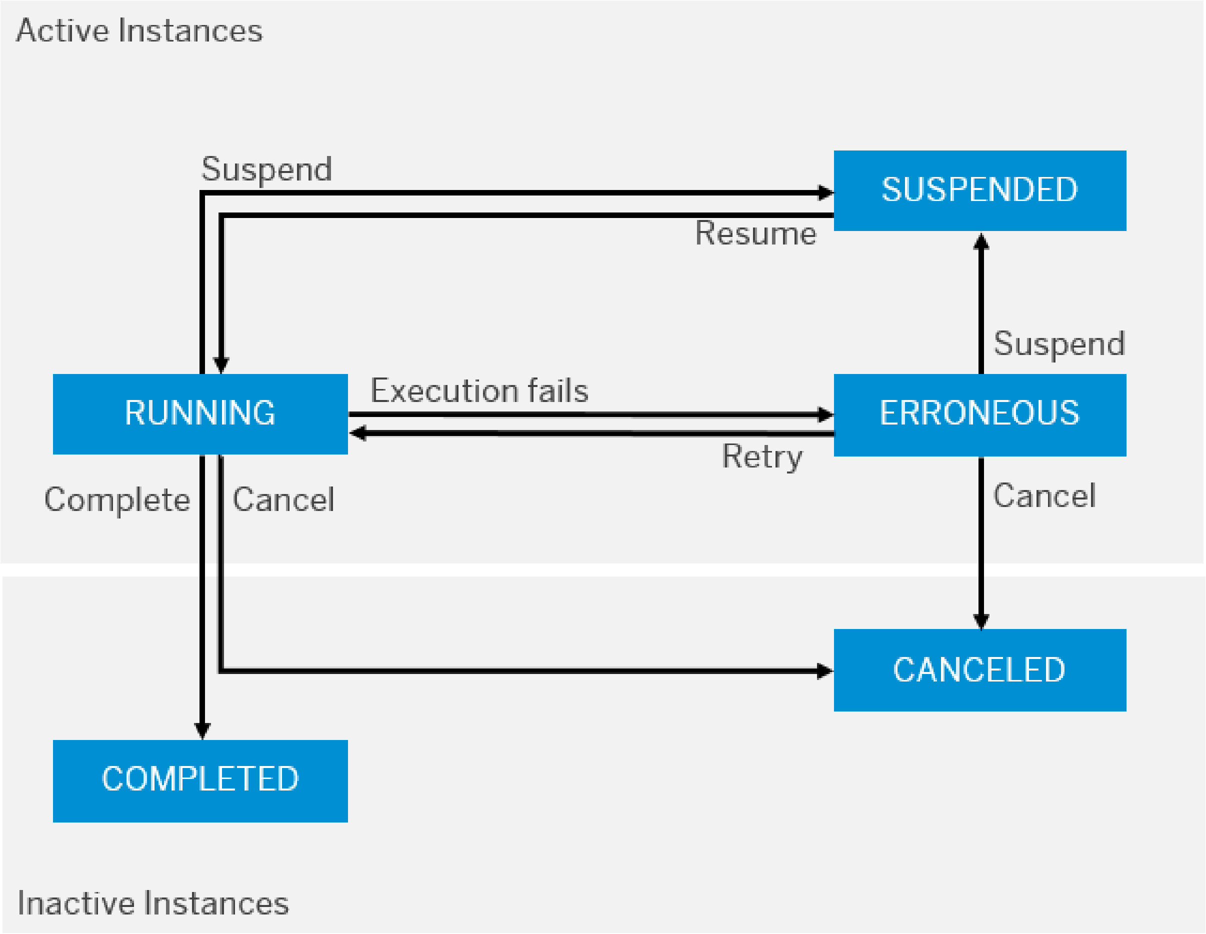
If the execution of a workflow element fails, it is retried several times. After that, the workflow element is kept but is not executed again. The workflow instance then changes to the ERRONEOUS status. However, only the execution branches with failed workflow element executions are affected. Parallel 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.

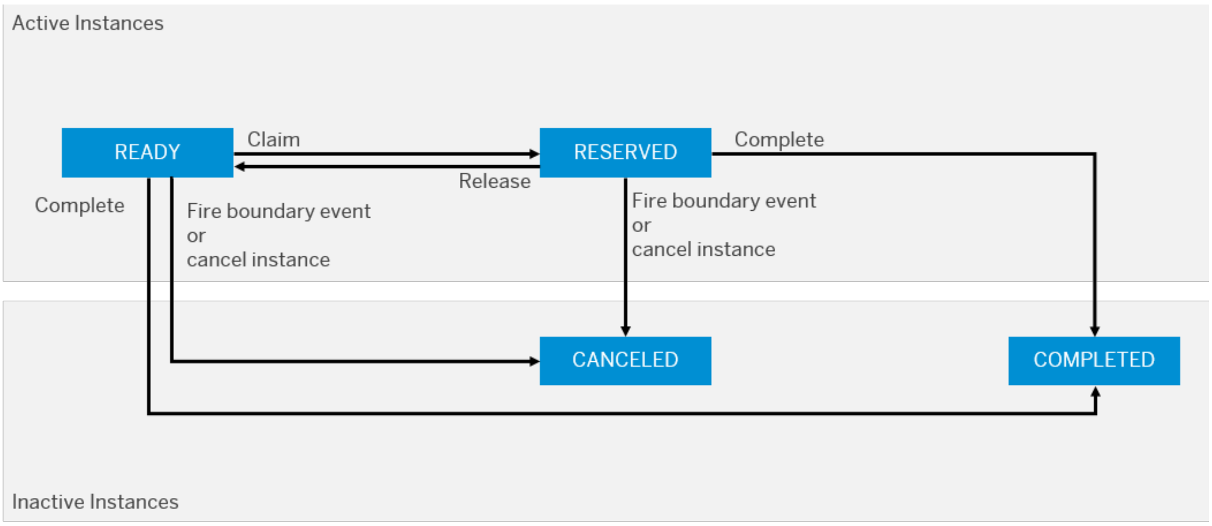
When a recipient claims the task, its status changes to RESERVED. When the user releases the task again, its status reverts back to READY.

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 278].

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 capability.

Consider this information during development to help you to achieve optimal use of the service.

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

Execution Limits

Area	Limit	Value for Standard Plan (Paid Account)	More Information
Workflow context	Size of the workflow context	100 KB per workflow instance	- 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. - See Creating and Reading Workflow Context Structures [page 54].
Workflow deployment	Size of the content in each workflow module in the MTA	15 MB	n.a.

Area	Limit	Value for Standard Plan (Paid Account)	More Information
API	Request rate limit	150 requests per second and tenant	• Even under optimal conditions, for example, in read-only scenarios, the request rate is limited to the indicated value. Depending on the scenario, resource limitations reduce the request rate that can be achieved. • 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. • Concurrent processing initiated by API requests, for example, workflow starts, can effectively reduce the success rate of the calls. • See Using Workflow APIs [page 156].
	Request body size	512,000 bytes	n.a.
	Processing time	30 seconds	Includes response generation by the server.
Script tasks	Execution time	150 milliseconds	See Configure Script Tasks [page 52].
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 about how to implement service tasks if high execution times are common, see Configure Service Tasks [page 46].
	Response size	100 KB	Together all service task responses stored in the workflow context must not exceed the workflow context limit.
Deployments	Number of deployments per tenant	-	Including the number of versions of all workflow definitions.
Referenced subflow	Nesting level of referenced subflows	20	Limits the nesting level of a referenced subflow.

Area	Limit	Value for Standard Plan (Paid Account)	More Information
Workflow instances	Number of workflow instances	-	All workflow instances in a tenant, regardless of the status.
Workflow activities within a workflow instance	Number of workflow activities that can be executed before the system interrupts further execution	2000	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 the next line "Number of retries available".
	Number of retries available after the system has interrupted the execution	10	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	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 SAPUI5 version 1.71 (latest patch) or higher.
    - To render workflow forms, you need SAPUI5 version 1.71 (latest patch) or higher.
- Pagination is not supported for My Inbox in Expert View. The maximum number of tasks, displayed by My Inbox Expert View is 1000. For more information, see *GET:/TaskCollection* in Inbox API for Cloud Foundry.
- Mass actions are not supported. You cannot process multiple tasks at one go in My Inbox.
- My Inbox displays tasks from the local SAP BTP Workflow tenant only. Federation from other task providers is not possible.

- My Inbox supports only SAPUI5 UIs without internal (hash-based) navigation (see Routing and Navigation). UIs with hash-based navigation are not compatible with My Inbox.
- 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 list or a radio button control can have at most 100 selectable values.
  - For task forms, 100 attachments at most 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, 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 supported by the duration specification 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 instances that are directly created in a subaccount and those created while subscribing to an application that uses the workflow, share the content in the same subaccount. So, 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 maximum 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 evaluates the content type of a response and only parses content that was reported to be a JSON string. 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.

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, the only type supported is string. 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
	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, the only type supported is string. 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)	50000 characters

ⓘ Note

In accordance with RFC 8259, “Number” types in JSON data may be treated as IEEE 754 binary64 (double precision), with the respective limitations regarding precision and range.

ⓘ Note

The above limits apply both at runtime as well as in the model. If validation of the model already statically determines that the limit will be exceeded at runtime, deploying such a respective model is blocked. If such validation cannot determine statically that the limit is exceeded, it is executed at runtime and can lead to failed task instances. Note that some of the related issues cannot be remedied in this case. Therefore, appropriate care needs to be taken to ensure the limits are not exceeded at runtime.

1.5 Supported Languages

The workflow 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
  - Bulgarian
  - Catalan
  - Chinese (Simplified)
  - Croatian
  - Czech
  - Danish
  - Dutch
  - English (UK)
  - English (US)
  - French
  - German
  - Hebrew
  - Hindi
  - Hungarian
  - Indonesian
  - Italian
  - Japanese
  - Kazakh
  - Korean
  - Lithuanian
  - Malay

    - Norwegian
    - Polish
    - Portuguese (Brazil)
    - Russian
    - Serbian (Serbia)
    - Slovak
    - Slovenian
    - Spanish (Mexiko)
    - Spanish (Spain)
    - Turkish
    - Ukrainian
- 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 78]

1.6 Browser Support

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

> ⓘ 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
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 workflow capability.

Prerequisites

- You have either a global account that is entitled to use the workflow capability.
  For more information, see Get a Global Account.
- 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 Orgs Administration Using the Cockpit and Org Administration Using the Cloud Foundry CLI.
- 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 Role Collections and Roles in Global Accounts, Directories, and Subaccounts and Assign Role Collections.
  For more information, see Managing Member Authorizations in the Neo Environment and Roles and Role Collections.

Using the Automatic Setup

You set up the workflow capability for your enterprise global account using a booster.

Procedure

1. Access your global account page in the SAP BTP cockpit, and choose *Boosters* from the left-hand navigation area.
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 does the following:

   - Sets up the cockpit for all SAP Workflow Management capabilities: Workflow, Business Rules, Process Visibility, and Workflow Management.
   - Assigns entitlements and quota for: workflow capability, business rules capability within the SAP Workflow Management service, process visibility capability within the SAP Workflow Management service, and SAP Workflow Management.

> ⓘ Note
>
> The workflow capability is added with the *standard* service plan.

- 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, business rules, process visibility, workflow management service.
- Creates the following destinations: BUSINESSRULES_APIHUB, BUSINESS_RULES, WM_CF_SPACE_PROVIDER, SAP_API_Business_Hub, `bpmprocessvisibility`, `bpmrulesruntime`, `bpmworkflowmanagement`, `bpmworkflowruntime`, and `bpmworkflowruntimeoauth`.
- Assigns all role collections of SAP Workflow Management to the user.

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 the Workflow Capability [page 190]

Using the Manual Setup

You can set up the workflow capability 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 *standard* service plan of the workflow capability to your subaccount.

   For more information, see Configure Entitlements and Quotas for Subaccounts.
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 and Roles.
   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 Create a Subaccount

3. To enable developers to develop applications with the workflow capability, assign the `WorkflowDeveloper` role. For more information, see Authorization Configuration [page 258].

Related Information

Assign Workflow Roles to Your Users [page 19]
Configuring the Workflow Capability [page 190]
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. Create a role collection. See Define a Role Collection.
2. Add roles to the role collection. See Add Roles to a Role Collection
3. Assign the role collection to a group of users. See Assign User Groups to Role Collections.

Related Information

Authorization Configuration [page 258]

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 Module* 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 Capability

Developer tasks for the workflow capability that are executed in the workflow editor or in the workflow runtime.

Related Information

Introduction [page 5]
Modeling a Workflow [page 21]
Creating User Interfaces [page 111]
Build and Deploy the Workflow Module [page 80]
Using Workflow APIs [page 156]

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.

> ⓘ Note
>
> The workflow editor does not support the 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:

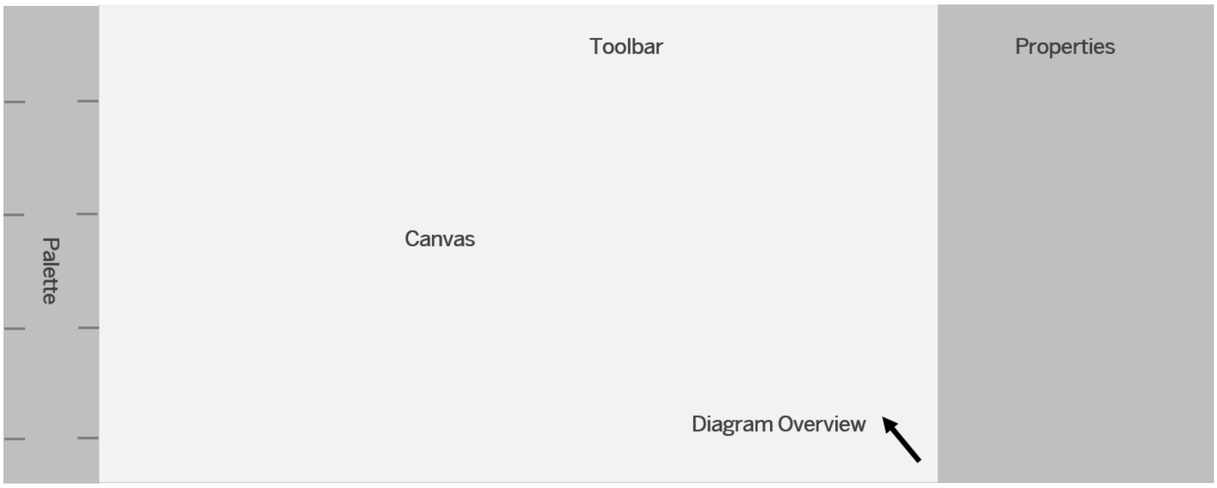
- Defining a start point of the workflow: Define a start point of the workflow using the start event. For more information, see Events [page 69].
- Defining workflow steps and their sequence: Define the process steps using the following graphical objects:
  - Tasks: There are user tasks that are performed by a human or a mail program, and 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 73].
- Defining an endpoint of the process: Defines an endpoint of the process using an end event or a terminate end event. For more information, see Events [page 69].

Related Information

Create a Workflow Module [page 23]
Legacy: Enable the Workflow Editor in SAP Web IDE [page 91]
Model Workflows in a Multitarget Application Project with SAP Web IDE [page 91]
Build and Deploy the Workflow Module [page 80]
Accelerated Modeling with Speed Buttons [page 82]
Expressions [page 83]
Migrate Workflow Project to a Multitarget Application Project with SAP Web IDE [page 109]
SAP Business Application Studio - Developer Guide

3.1.1 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 dragging the required flow object in the palette to the canvas.
- 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. You can also use the diagram overview to navigate to the required part of the workflow.

3.1.2 Create a Workflow Module

Create a multitarget application (MTA) project that contains a workflow module.

Prerequisites

You have created a dev space with the following extensions:

- You have the SAP Predefined Extension *MTA Tools* that comes, for example, with the *SAP Fiori* or *SAP Cloud Business Application* application types.
- You have the SAP Predefined Extension *Workflow Module*.

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

Context

The generated module references the SAP Build Process Automation service instance and no longer the SAP Workflow Management service instance. If you want to use the module in SAP Workflow Management, see Adapt Workflow Module for Use with Workflow Management Service Instance [page 24].

Procedure

1. To model workflows as part of an MTA, open your dev space: From the menu, choose *File* > *Open Workspace*.
2. Select your folder of choice, for example, *projects*, then choose *Open*.

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

3. If you do not have an MTA in your dev space that you want to use here, do the following to create one.
   a. Open *File* > *New Project from Template*.
   b. Select *Basic Multitarget Application*.
   c. Enter a project name, and then choose *Finish*.

   The IDE opens again with MTA as a workspace.
4. Right-click the `mta.yaml` file, and choose *Create MTA Module from Template*.
5. Choose *Workflow Module*, and then choose *Start*.
6. Enter a path to the location where the module is to be generated. We recommend using the proposed path.
7. Enter a workflow module name.
8. Choose *Next*.

9. Enter a workflow name, a namespace, and a description. Then choose *Finish*.

   The workflow module, together with a workflow, is created and opens.

3.1.2.1 Adapt Workflow Module for Use with Workflow Management Service Instance

All new workflow modules generate a reference to the SAP Build Process Automation service instance instead of to the SAP Workflow Management one.

Context

If you want to deploy the workflow module using an SAP Workflow Management service instance, you must adapt the generated `mta.yaml` file.

Procedure

1. In SAP Business Application Studio, open the `mta.yaml` file that you have generated as described in Create a Workflow Module [page 23].

   It looks similar to this sample:

   Sample Code

   ```
   modules:
   - name: workflow-module
     type: com.sap.application.content
     path: workflow-module
     requires:
     - name: sap_processautomation
       parameters:
         content-target: true
         service-key:
           config:
             deployUsageScenario: workflow
           name: spa-workflow-service-key
   resources:
   - name: sap_processautomation
     type: org.cloudfoundry.managed-service
     parameters:
       service: process-automation-service
       service-plan: standard
   ```

2. Update the require and resources sections of the `mta.yaml` file as follows:

   Sample Code

   ```
     requires:
     - name: workflow
   ```

```
    parameters:
      content-target: true
resources:
- name: workflow
  type: org.cloudfoundry.managed-service
  parameters:
    service: workflow
    service-plan: standard
```

3. Save your changes.

3.1.3 Define Workflows

Use this procedure to define workflows.

Procedure

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

   For more information, see Open a Workflow [page 28].
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* such as "**`Employee onboarding process initiated for ${context.employeename}`**". For more information, see Expressions [page 83].

   > Note
   >
   > - For more information about character limits for the workflow capability, 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 156].
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 the workflow capability, 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 in the *Workflow Properties* pane to add the attributes for a workflow. To configure these attributes, see Configure Custom Workflow Attributes [page 76].
5. To model the start event of a workflow, select *Events* (○) and then *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 read-only.

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 when you start a new workflow instance. For more information, see Configure Start Events [page 69].

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

   For more information, see Workflow Capability API
8. To add a task to the workflow, see Tasks [page 28].
9. To add a gateway to the workflow, see Gateways [page 73].
10. To add an intermediate message event, see Configure Intermediate Message Events [page 70].
11. To add an intermediate timer event, see Configure Intermediate Timer Events [page 72].
12. To connect two flow elements, choose the ➔ icon.

    > ### ⓘ Note
    >
    > If you choose a flow element using the speed buttons, 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 82].
13. To model the end event of a workflow, choose *Events* (○) and then *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* (○) and then *Terminate End Event* (◉) and drop it onto the canvas from the palette.

    For more information about the terminate end event, see Events [page 69].

    > ⓘ Note
    >
    > You can also model a terminate end event using the speed buttons. For more information, see Accelerated Modeling with Speed Buttons [page 82].

16. In the *End Event Properties* pane of the terminate end event, 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
    >
    > Each time you change the properties of flow elements, press ENTER to enforce the update of the property.

3.1.3.1 Create a Workflow

Create a workflow from scratch within an existing workflow module.

Prerequisites

You have created a workflow module. See Create a Workflow Module [page 23].

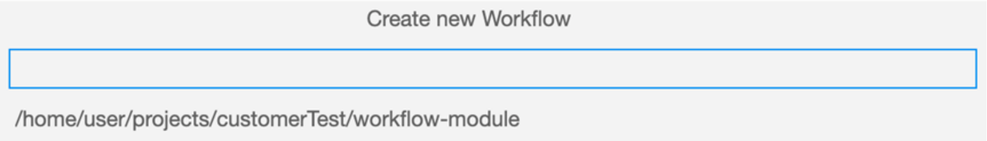
Procedure

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

    > ⓘ Note
    >
    > Make sure to enter the complete path and not just the name of the workflow. Check whether you can use the proposed path.

    Example: `/home/user/projects/customerTest/workflow-module`

    At a later step in this wizard, you are prompted for the workflow name.



Create new Workflow

/home/user/projects/customerTest/workflow-module

3. Enter the required details and confirm.

   The workflow editor opens.

   > ⓘ Note
   >
   > Deploying 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]. We recommended that you specify a namespace for your workflow. The namespace is generated automatically into the workflow ID.

3.1.3.2 Open a Workflow

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.

   > ⓘ Note
   >
   > SAP Business Application Studio loads the extensions from a central repository lazily. Depending on your network connection, the editor might not be ready yet and a text editor is shown instead. Wait a bit, and 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 285].

3.1.3.3 Tasks

The workflow capability 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 46]
Configure Script Tasks [page 52]
Configure Mail Tasks [page 62]

3.1.3.3.1 Configure User Tasks

Use this procedure when you want a user to perform a particular task in the workflow.

Procedure

1. Choose ▭ (Tasks), then *User Task* from the palette and drop it onto 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.

   > ⓘ Note
   >
   > - For more information about character limits for the workflow capability, see Conventions, Restrictions, and Limits [page 8].
   > - A unique *ID* is generated for every workflow artifact. This ID is read-only.
   > - Make sure 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 the *User Task Properties* area, choose the *Details* tab.
6. Depending on the priority of the user task, choose one from the *Priority* menu.
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, e-mail address or logon names. See also Guidelines for Specifying Recipient Users [page 45].

   > ⓘ 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 a dynamic

context, you can specify the expression in the *Subject* field as "**`Approval for ${context.employee.name}`**". For more information, see Expressions [page 83].
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*.
  You can either use groups defined using the Identity Authentication service or you can enter a role collection created with the SAP Authorization and Trust Management service (technical name: xsuaa). For more information, see:
  - User Groups in the Identity Authentication documentation
  - Create an XSUAA Instance
- 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 or due date by when the task is due, select the *Configure Due Date* checkbox.

   Configure the due date using one of the following options:

   - Duration: You have the following options to configure this field:
     - To provide a duration for the due date as an expression, choose *Expression* from the *Due Date Based On* dropdown list. Now, provide the due date in the *Duration* field as an expression.

       ⓘ Note

       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.

     - 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.
   - Timestamp (absolute time)
     Define the timestamp using a single JUEL expression that must be translated to the absolute time in the ISO 8601 format at runtime. Example: ${context.dueDate}
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 Inbox API for Cloud Foundry, select *Allow Forward*.

ⓘ Note

- A forwarded user does not need to be a configured recipient. However, this user becomes one of the task recipients so 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 must 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 209] and Legacy: Create Workflow and My Inbox Tiles on SAP Fiori Launchpad [page 219].

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 82]
Configure a Custom Task User Interface Using an HTML5 App [page 33]
Configure Boundary Timer Events [page 31]
Conventions, Restrictions, and Limits [page 8]

3.1.3.3.1.1 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.

Context

Boundary timer events are attached to a user task. Some user tasks must be completed during a certain time interval. You can add a boundary timer event to define the time until when 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 buddy is responsible for confirming the equipment that needs to be procured for the new hire.

> A noncanceling 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 the 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. Therefore, 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. Configure the timer using one of the following options.
    - Duration (date field relative to the task creation time)
      You have the following options for configuring this field:
        - To use expressions, choose *Expression* from the *Duration Based On* dropdown list.

          > ⓘ Note
          >
          > Provide an expression in the *Duration* field using a subset of the ISO 8601. For example, **`PT${context.minutes}M`**. The Java Unified Expression Language (JUEL) expression **`${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 list. 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 list.

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

    - Timestamp (absolute time)
      Define the timestamp using a single JUEL expression that must be translated to the absolute time in the ISO 8601 format at runtime. Example: **`${context.paymentDueDate}`**
5. To define the boundary timer event as a canceling event, select the *Cancel Task* checkbox.
6. Choose *Save*.

ⓘ Note

- You can add multiple boundary timer events to a user task, which get 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.
- Pay special attention when 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.

3.1.3.3.1.2 Configure a Custom Task User Interface Using an HTML5 App

With the workflow capability, end users can access their workflow tasks in their inboxes using 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 112]. To use an HTML5 application for user tasks, it needs to first 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.

    ⓘ 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*.

> ⓘ Note
>
> *SAPUI5 Component* is added automatically but can be edited. If the selected HTML5 application is configured for a managed approuter, then the *Business Solution Name* is also added automatically. You can edit that entry 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. ⓘ Note The value 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, let's say a task UI contains three actions: 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 115].

> ⓘ Note
>
> - You can enter multiple key value pairs separated by commas. 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
> - Consider the following when 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 83].
> - Expressions are evaluated when the task is created, that is, they cannot be used to transfer data to the user interface that changes after this time. For such cases, see Legacy: Set the Task and Task Context Models [page 125].

4. Save the workflow.

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

Context

> ⓘ 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. 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.

    > ⓘ 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*.

ⓘ Note

SAPUI5 Component is added automatically, but you can edit it.

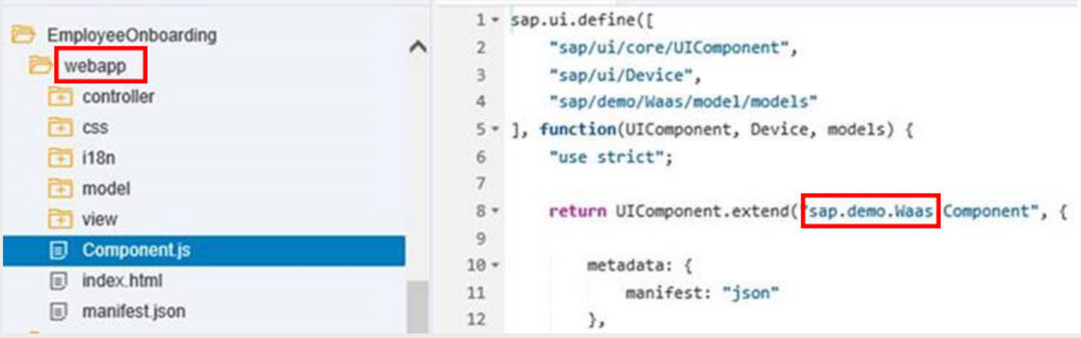
2. To manually provide the custom task UI details, provide following details in the *User Interface* tab:

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 these *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. See the sample screenshot and the corresponding *User Interface* properties in the following table:

  Example

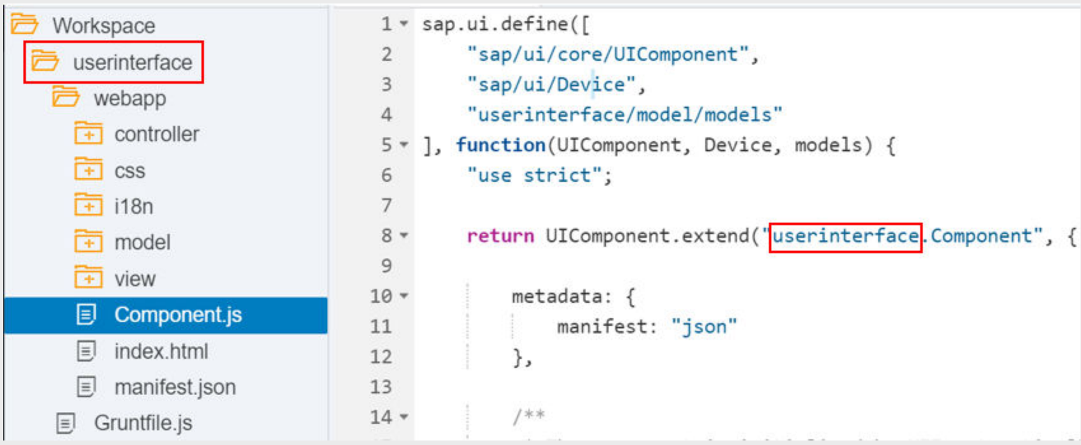
  

```
sap.ui.define([
    "sap/ui/core/UIComponent",
    "sap/ui/Device",
    "sap/demo/Waas/model/models"
], function(UIComponent, Device, models) {
    "use strict";

    return UIComponent.extend("sap.demo.Waas.Component", {

        metadata: {
            manifest: "json"
        },
```

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 in the following example:

Example



```
sap.ui.define([
    "sap/ui/core/UIComponent",
    "sap/ui/Device",
    "userinterface/model/models"
], function(UIComponent, Device, models) {
    "use strict";

    return UIComponent.extend("userinterface.Component", {

        metadata: {
            manifest: "json"
        },

        /**
```

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 SAPUI5 component needs to be used for different task UIs with minor modifications, the URL parameter can be used to define the modification.

   For example, let's say a task UI contains three actions: 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. Task UI developers can then access these parameters in their custom task UI and choose to show or hide specific actions. For more information, see Access the User Task Data [page 115].

   Note

   - You can enter multiple key value pairs separated by commas. 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
   - Consider the following when 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 83].
- Expressions are evaluated when the task is created, that is, they cannot be used to transfer data to the user interface that changes after this time. For such cases, refer to Legacy: Set the Task and Task Context Models [page 125].

4. Save the workflow.

Related Information

Configure User Tasks [page 29]

3.1.3.3.1.4 Configure a User Task UI Using Workflow Forms

With the workflow capability, end users can access their workflow tasks in their inboxes using 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 136]. To use a task form for user tasks, you need to configure it.

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*
      In the *New Form* dialog, enter the following data:

Field	Description
Name	Name of the form that you are creating
ID	Identifier of the new form

Field	Description
Revision	Revision of the form For more information, see Versioning Forms [page 152].

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

- *Select*
  In 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 152].

For more information about forms, see Creating a Workflow Form [page 136].

4. Save the workflow.

   The form is created in a separate *forms* folder in the workflow module in a folder with the same name as the workflow for which the form is created.

Related Information

Configure User Tasks [page 29]
Creating a Workflow Form [page 136]

3.1.3.3.1.5 Configure Custom Task Attributes

You can assign custom task attributes to user tasks.

Context

With custom task attributes, such as project ID or project name, you can define business-related properties and assign them to user tasks. The business-related properties are assigned, using JUEL expressions, when a task is created.

At runtime, you can use the respective the workflow capability 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 156].

Procedure

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

> ⓘ 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. ⓘ Note You can only use alphanumeric characters (a-z, A-Z, 0-9) and the underscore character for the ID. The ID must begin with an alphabet character or the underscore but not a number.
Label	A human readable name of the attribute, which appropriate user interfaces can use to label the attribute.
Type	Data type of the attribute. ⓘ Note Currently, only the string data type 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.

> ⓘ 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]

3.1.3.3.1.5.1 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 contextual workflow data can help business users work more efficiently with My Inbox.

For more information, see Custom Attributes in My Inbox [page 252].

> ⓘ Note
>
> This feature is disabled by default 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 an administrator

The custom attributes of the these types are visualized in the 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

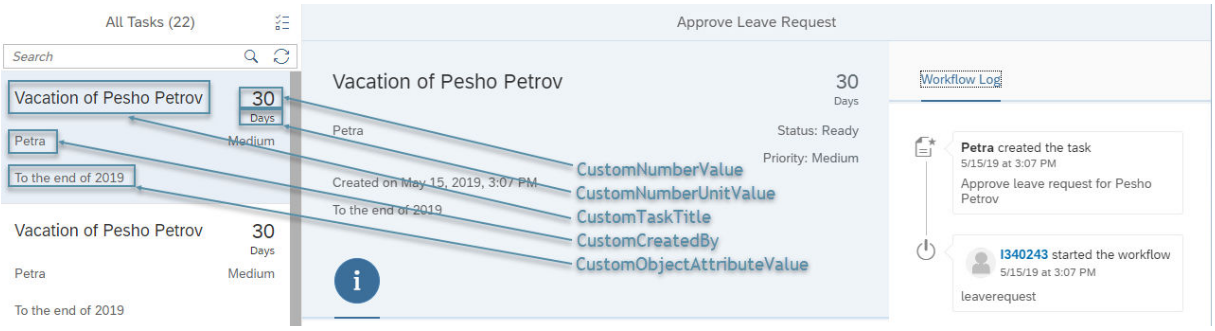
> ⓘ Note
>
> You cannot sort or filter by these values. In addition, they are not visualized as columns in the *Expert View* of My Inbox.

The following table shows you 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	**Use**	**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	☑		☑
CustomObjectAttributeValue	Displays additional contextual information about the task	☑		☑

		Visualization		
Custom Attribute ID	**Use**	**Master-Detail List**	**Expert View**	**Generic UI for Task Details**
CustomCreatedBy	Replaces the name of the task creator with a custom value	☑	☑	☑

The following screenshot shows you where the custom attributes with SAP Reserved IDs are visualized in the *Master-Detail View*.



Vacation of Pesho Petrov 30 Days
Petra Medium
To the end of 2019

Vacation of Pesho Petrov 30 Days
Petra Medium
To the end of 2019

Vacation of Pesho Petrov 30 Days
Petra
Status: Ready
Priority: Medium
Created on May 15, 2019, 3:07 PM
To the end of 2019

Petra created the task
5/15/19 at 3:07 PM
Approve leave request for Pesho Petrov

I340243 started the workflow
5/15/19 at 3:07 PM
leaverequest

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 an 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 the *Master-Detail View* and the *Expert View* of My Inbox. In addition, they can be exposed as columns in the *Expert View* of My Inbox.

They allow you to:

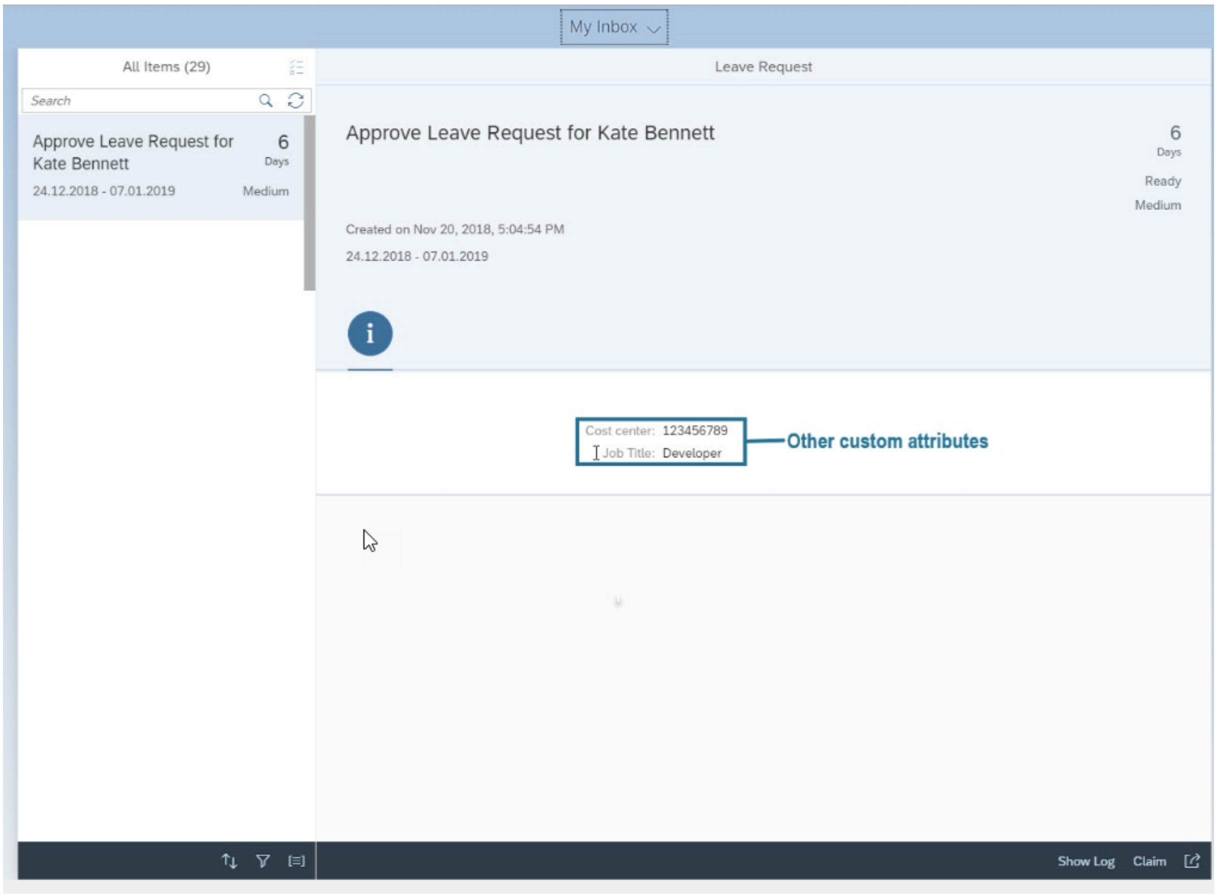
- Expose additional information about the task in the *Information* tab of the generic UI for task details in My Inbox.
- Expose the custom attributes as columns in the *Expert View* of My Inbox. See Expert View [page 246].
- Enable business users to quickly find tasks by sorting and filtering based on custom attributes in the *Expert View*.

Procedure

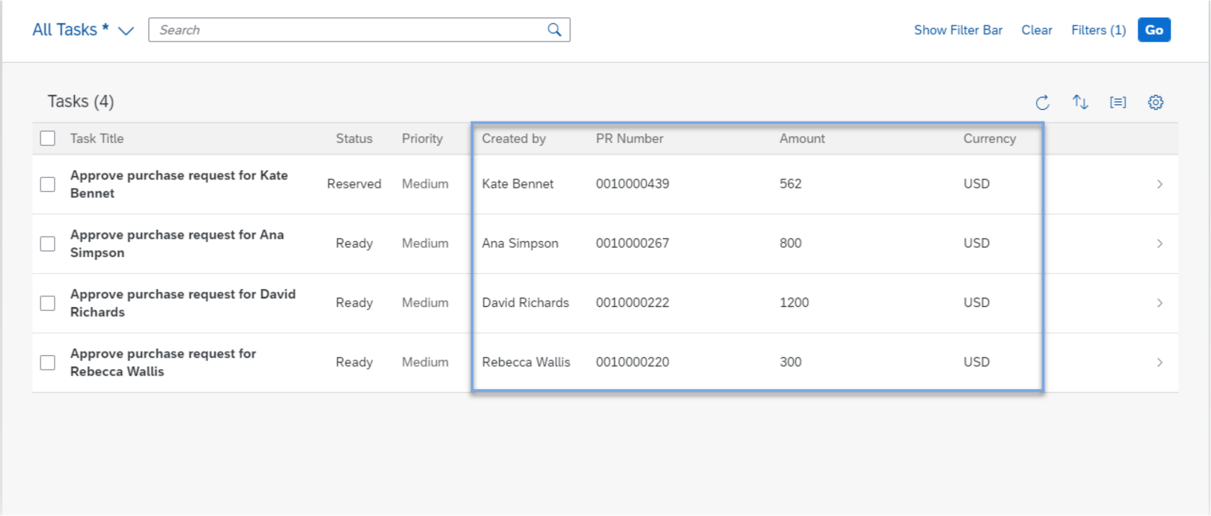
- For the ID of a custom attribute, assign a random ID that is different from the SAP Reserved IDs for custom attributes.

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

> ⓘ Note
>
> Sorting and filtering are not possible based on this custom attribute data in the *Master-Detail View*. You can use the *Expert View*instead.



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



Tasks (4)

☐	Task Title	Status	Priority	Created by	PR Number	Amount	Currency
☐	Approve purchase request for Kate Bennet	Reserved	Medium	Kate Bennet	0010000439	562	USD
☐	Approve purchase request for Ana Simpson	Ready	Medium	Ana Simpson	0010000267	800	USD
☐	Approve purchase request for David Richards	Ready	Medium	David Richards	0010000222	1200	USD
☐	Approve purchase request for Rebecca Wallis	Ready	Medium	Rebecca Wallis	0010000220	300	USD

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

> ⓘ Note
>
> If the tasks use a custom task UI, the custom attributes are not visualized as part of the custom task UI.

> ⓘ 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 156].

3.1.3.3.1.6 Guidelines for Specifying Recipient Users

When you specify recipient users for a user task, consider the following:

> ⓘ Note
>
> Carefully consider the impact that the changes described here might have on your overall scenario. Changing certain settings after productive use has started can have a negative impact on scenarios that are incompatible with the change. If applicable, use mechanisms that restrict the impact to the specific scenario.

- Evaluate whether you can use “recipient groups” instead of “recipient users” because there are limits as to how many recipient users may be specified, see Conventions, Restrictions, and Limits [page 8]. If you can, you must configure the assignment of users to groups in the identity management-related function of the platform or the identity management back-end systems. This has, for example, the benefit that the assignment of a task to a certain user can be removed using these central identity management functions instead of in the workflow definition and related locations. This usually improves compliance with company and legal requirements. For example, removing an assignment typically becomes effective as soon as authentication tokens expire.

- If you cannot avoid specifying user names using constants or expressions, make sure that you apply the necessary lifecycle actions on the respective events to achieve compliance. For example, use the administrative REST APIs of the workflow capability, to remove recipient users when they should no longer be assigned to a task because they left their department or the company. See Using Workflow APIs [page 156]. Also, ensure that user interfaces that allow configuration of user IDs apply appropriate validation on the task recipients.
- Ensure that the case and spelling of the user ID matches the respective fields of the authentication tokens exactly. It is important that lower or upper case is also considered, because the workflow capability matches them as is. The workflow capability must also consider case sensitivity for user names that look like email addresses. There is no metadata that indicates whether user names are actual email addresses or whether case sensitivity is irrelevant. For this purpose, check your identity management system and the related configuration of SAP Business Technology Platform.
- Check the *User ID Source* and its related settings. Consider using “E-Mail” as the configured value, because this might improve consistency of user names in a scenario.
- Ensure that the identifiers, as validated against the user database, are provided. Do not rely, for example, on user names as entered directly by the user.
- Evaluate whether you can disable the creation of “shadow users”. In certain constellations, this prevents users from logging in with user names that do not correspond to the canonical identifier, but use a different case.
- Evaluate whether you can configure that user inputs are automatically converted to the expected case. If you can, see the documentation of the SAP Cloud Identity Services feature *Apply Function to Subject Name Identifier* in Convert Subject Name Identifier to Uppercase or Lowercase or the respective configuration of your custom identity management system.
- To restrict the impact of these changes, you can implement certain mitigations using the service task or script task features of the workflow capability. With these, you can validate or fix the case of user identifiers using the respective validation REST services. These might be available from the identity management system or can be implemented as a separate service.

3.1.3.3.2 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 capability therefore have the purpose of freeing up resources as early as possible for other tasks.

> → Tip
>
> We recommend configuring the service execution time to be 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 that returns quickly. It can report back the execution result using message events. For more information, see Configure Intermediate Message Events [page 70].

The workflow capability 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 of 200 to 299. Ensure that these retries can complete successfully, for example, after administrative intervention and even if there are 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 capability might fail permanently or create duplicate entities.

Certain workflow capability REST APIs aren't idempotent and shouldn't be called to modify the currently running workflow instance. For more information and recommendations, see SAP Note 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 that 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 262].
   - Destination can also refer to the dynamic workflow context. For more information, see Expressions [page 83].
   - For more information about character limits for workflow capability, see Conventions, Restrictions, and Limits [page 8].
7. Select one of the following options from the *Choose a Service From* list:
   - *SAP Business Accelerator Hub*
   - *Others* (default)

   SAP Business Accelerator 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 Business Accelerator Hub.
8. If you selected *SAP Business Accelerator Hub*, perform the following procedure Configure a Service from SAP Business Accelerator Hub [page 51].
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 example below.
- Services that are called from a service task must support the JSON format for request and response body. Consequently, the workflow capability sends the `Accept: application/json` header in every HTTP request, and expects the service to return `Content-Type: application/json`. Other responses are declined by the workflow runtime, which can lead to a runtime error.
- Ensure the URL that is concatenated from the *Destination* and the *Path* are valid.
- The workflow capability runtime ensures proper encoding of the final URL that is invoked. To avoid double encoding, don't enter the URL specified at the destination, the value for the path property, and the `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. An XSRF token is used for modifying operations that are protected against XSRF (cross-site request forgery) attacks. For more information, see Protection from Cross-Site Request Forgery.

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. For more information and further guidance about the CSRF token mechanism, see the support portal.

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.

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 `${context}` can't be used in the response variable. You must specify a variable within the context to be used as a response variable. For example, `${context.leaveRequest}` or `${context.leaveRequest.response}` are valid response variables.

> Example
>
> This example shows how to call a REST service to store employee 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 this:

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 198].

    b. In the *Flow Element* section, choose *Select*.

    This field is only available 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 50].

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 capability 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 262].

Related Information

Accelerated Modeling with Speed Buttons [page 82]
Connectivity Options Blog

3.1.3.3.2.1 Model Header Parameters in the Workflow Editor

HTTP header parameters are a powerful way to transmit additional information between a workflow and an external system with the submitted request. For a workflow, prominent examples are requesting a specific JSON-like content type (in range of application/*+json) or content language.

Context

For more information about HTTP headers, see https://developer.mozilla.org/docs/Web/HTTP/Headers .

> ⓘ Note
>
> There are cases, in which the headers that 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*.

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. If the JUEL expression can't be resolved, the unresolved JUEL expression is passed.
4. To remove a row, first select it, then choose *Delete*.
5. Save your changes.

3.1.3.3.2.2 Configure a Service from SAP Business Accelerator Hub

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

Prerequisites

When using SAP Business Application Studio, SAP Business Accelerator Hub is set as a destination. See Configure an SAP Business Accelerator Hub Destination [page 208].

Procedure

1. In the *Service* section, choose *Select* to browse for an 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*.

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

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

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

8. Choose *Finish*.

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

3.1.3.3.3 Configure Script Tasks

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

Context

With script tasks, you can create, read, and modify workflow context structures to transform data between different representations and to prepare data as required for other tasks, for example, service tasks (request variable). You can access contextual information about the workflow instance or from user task instances in the workflow instance. You can enrich the transformations with information or identifiers from the workflow instance. You can define instance-specific authorization data, for example, based on responses from prior service tasks, complementing respective REST APIs.

Note that script tasks are subject to resource limits to protect against excessive usage. Due to the resource constraints, transformations may exceed the supported complexity of script tasks. In such cases, reduce the complexity of transformations, for example, by using more compact input data or incremental transformations. You should also consider alternative means for transformations, for example, using service tasks to call external services. Apply common ECMAScript code optimizations to the scripts, for example:

- Reduce the size of the data processed in the scripts, for example, through early data cleanup.
- Avoid regular expressions that are equivalent to simple string comparisons.
- Store data and workflow API references (properties of the $ object) in intermediate variables when accessing them multiple times.

> → Recommendation
>
> We recommend that you export and import workflow projects, rather than individual workflows, when 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.

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

3. To add JavaScript files, choose *Select* to browse for the JavaScript file in the current project, or perform the following steps to create a new file:
   a. Choose the *Create File* link.
   b. In the *Create New File* window, provide a file name.
   c. Choose *Create*.
   d. In the JavaScript file, provide the script.
4. Save the workflow.

Results

> ⓘ Note
>
> - To view and edit the JavaScript file, select 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 54]
>   - Accessing Contextual Information During Execution of Script Tasks [page 58]
> - 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'` aren't supported for script tasks. Using the `function` keyword is supported, but you can't 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 capability have the purpose of freeing up resources as early as possible for other tasks. These limits protect against excessive usage, for example, caused by inefficient 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].

Related Information

Accelerated Modeling with Speed Buttons [page 82]

Conventions, Restrictions, and Limits [page 8]

3.1.3.3.3.1 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

ⓘ 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 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*.

⚠ Caution

Data formats that are not natively supported by JavaScript might get converted. That is, integer or long numbers might get converted to a (double) number depending on their size and other factors. A certain representation cannot be guaranteed. This might break implementations that rely on a certain number precision or representation.

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 56]

Modifying the Workflow Context with Arrays [page 57]

3.1.3.3.3.1.1 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"
}
```

3.1.3.3.3.1.2 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
```

3.1.3.3.3.2 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	-

Property Name	Type in Script Task	Comments
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

* Dates are represented as strings in expressions. For more information, see Expressions [page 83].

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

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

Script tasks cannot modify the `$.usertasks` API. All its properties are provided by workflow capability 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.

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 the 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;
```

3.1.3.3.3.3 Get and Set Instance-Specific Roles

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

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

3.1.3.3.4 Configure Mail Tasks

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

Prerequisites

You have configured a mail destination. See Configure the Workflow Capability Mail Destination [page 199].

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 the *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 the workflow capability, see Conventions, Restrictions, and Limits [page 8].

6. (Optional) Add mail addresses to the *Cc* 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, the mail task fails when at least one invalid recipient is determined.

8. Provide a *Subject*.

> Note
>
> The *Subject*, *Cc*, *Bcc*, and *To* fields can contain JUEL expressions. For more information, see Expressions [page 83].
>
> 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 text form.
   - *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 mails 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 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 Management service</p>
    </body>
</html>
```

5. Save and close the HTML file.

> ⓘ 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]

3.1.3.3.5 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 that you use to 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 and then 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 workflow when the subflow ends.

The 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 main workflow that is calling the subflow.

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 (Referenced Subflow ⊞) 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 that you want to use as a subflow.

   Either type in the ID as text or use a JUEL expression such as `${context.propname}`.

   To avoid typing, choose *Select* (next to *No file selected*). Then expand the folder icon ▶ *<module name you defined>* ▶ *workflows* ▶ *<one of the displayed workflows>* ▶.

   You can only select workflows of the same module and not the same workflow itself.
5. Optional. To execute multiple instances of the referenced subflow in parallel, enter the following on the *DETAILS* tab:
   a. For the type, select *Parallel* under *Loop Configuration*.
   b. Enter JUEL expressions for the *Collection Context Path*, for example, `${context.items}`.

      For each item in the referenced collection, the subflow will be executed once.

      > ⓘ Note
      >
      > In the input and output mappings of the subflow, you can refer to individual items in the collection by using the `loop.counter` variable, for example, `${context.items[loop.counter].status}`.

   c. Optional. You can specify a *Completion Condition*, for example `${context.items[loop.counter].status == "REJECTED"}`. If this condition evaluates to true while a single instance of the parallel subflow is completed, the parallel execution is considered completed as well. All other parallel instances are canceled. Output mappings for such canceled instances are skipped. For more information regarding input and output mappings, see below.

      The subflow icon in the editor is updated ReferencedSubflow1 ||| ⊞.
6. 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.
7. Optional. To include a timer for the subflow, add a boundary timer event. For more information, see Configure Boundary Timer Events [page 66].
8. 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. You can use the following 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.

   > ⓘ Note
   >
   > To avoid ambiguities, automatic checks are run on the mappings to prevent you from defining overlapping target expressions between mappings. In this case, consider preparing the data by inserting a script task before the referenced subflow task.

Related Information

Referenced Subflow – Modularize your workflows

3.1.3.3.5.1 Configure Boundary Timer Events

Configure a boundary timer event to trigger an alternative flow if a referenced subflow doesn't finish within a specified time.

Context

Boundary timer events are attached to a referenced subflow. Some referenced subflows must be completed during a certain time interval. You can add a boundary timer event to define the time until when the flow can wait at the referenced subflow before starting an alternative flow. There are two types of boundary timer events:

- Canceling boundary event: When this event is triggered, it cancels the referenced subflow it is attached to.
- Non-canceling boundary event: When this activity is triggered, it does not cancel the referenced subflow it is attached to.

> ⓘ Note
>
> - You can add multiple boundary timer events to a referenced subflow, which gets triggered when the corresponding timers are fired. When a canceling boundary event is triggered, any boundary events attached to the same subflow that haven't been triggered yet are canceled.
> - Pay special attention when suspending and resuming a workflow instance with several boundary timer events on an active subflow. If such an instance is resumed and after it was suspended for a time period longer than the corresponding timer durations, then the events are triggered in a random order.
> - When you add multiple boundary timer events, they are placed on the same position at the bottom of the referenced subflow. This may lead to several events on top of each other. However, these events can be moved along the boundary of the referenced subflow.

Procedure

1. Choose *Boundary Timer* from the speed button of the required referenced subflow.
2. Provide a *Name* and *Documentation* for the boundary timer event.
3. In the *Boundary Timer Event Properties* area, choose the *Details* tab.
4. Configure the timer using one of the following options.
    - Duration (date field relative to the referenced subflow creation time)
      You have the following options to configure this field:
        - To use expressions, choose *Expression* from the *Duration Based On* dropdown list.

          > ⓘ Note
          >
          > You must provide an expression in the *Duration* field using a subset of 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 list. Now, provide the *Duration* as a numeric value, and choose a *Unit of Time*.
    - Timestamp (absolute time)
      Define the timestamp using a single JUEL expression that must be translated to the absolute time in ISO 8601 format at runtime. Example: **`${context.paymentDueDate}`**
5. To define the boundary timer event as canceling, select the *Cancel Subflow* checkbox.
6. Choose *Save*.

3.1.3.3.5.2 Configure Boundary Escalation Events

Configure a boundary escalation event to trigger an alternative flow based on an escalation event code.

Context

Boundary escalation events are attached to a referenced subflow. The referenced subflow instance must indicate the occurrence of an event to its parent workflow instance in the same workflow instances hierarchy. The throwing and catching of an event are connected using the escalation code. You can add a boundary escalation event on a referenced subflow in a workflow, define the escalation code, and model the alternative flow. There are two types of boundary escalation events:

- Canceling boundary escalation event: When this event is triggered, it cancels the referenced subflow it is attached to.
- Non-canceling boundary escalation event: When this event is triggered, it does not cancel the referenced subflow it is attached to.

Expected runtime behaviors:

- The escalation code thrown by a workflow instance can only be caught by exactly one parent workflow instance in the workflow instances hierarchy. When an intermediate escalation event is emitted, then the workflow engine searches for a boundary escalation event one level up in the calling instances hierarchy, that is, from sub-flow to the parent workflow. If it doesn't find a catching boundary escalation event, it goes up one level and continues until it finds one. Then, the workflow engine stops propagating up. If no catching boundary escalation event is found, the execution just continues with the normal flow.
- You can add multiple boundary escalation events, each catching different escalation codes to a referenced subflow.

  > ⓘ Note
  >
  > You can add several boundary events with the same escalation code to a referenced subflow. However, they are triggered in a random order.

- When you add multiple boundary escalation events, they are placed on the same position at the bottom of the referenced subflow. This may lead to several events on top of each other. However, these events can be moved along the boundary of the referenced subflow.

Procedure

1. Choose *Boundary Timer* from the speed button of the required referenced subflow.
2. Provide a *Name* and *Documentation* for the boundary escalation event.
3. In the *Boundary Escalation Event Properties* area, choose the *Details* tab.
4. Provide the escalation code to be thrown by the child workflow and to be caught in this workflow.

   > Example
   >
   > orderShipped, orderDelayed, approvalTaskRejected

> ⓘ Note
>
> JUEL expressions in the escalation code are not supported.

3.1.3.4 Events

An event affects the flow of the process.

The workflow capability editor supports the following events:

Start event: 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: Process steps where the respective workflow instance waits for a message before the flow commences in the respective control flow branch.

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

End event: An event with 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 are 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.

3.1.3.4.1 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 25].

Context

After the deployment of the workflow, the sample context is displayed in the *Monitor Workflows* app when you start a new workflow instance. For more information, see Workflow Definitions [page 281].

You can also use the API to retrieve the sample start context. For more information, see Using Workflow APIs [page 156].

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 46].

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, do the following:
   a. Choose the *Create file* link.
   b. In the *Create New File* window, provide a file name with a `.json` extension.
   c. Choose *Create*.
   d. In the JSON file, provide the context.

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

Related Information

Define Workflows [page 25]

3.1.3.4.2 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

You have configured a business key for your workflow. For more information about business keys, see Define Workflows [page 25].

Context

Clients can send messages using the REST endpoint. For more information about how to send messages, see the Workflow Capability API documentation in Using Workflow APIs [page 156].

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* (○) and then *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.

   For more information about character limits for the workflow capability, 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.

   > ⓘ 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 83].
   > - 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 a 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]

3.1.3.4.3 Configure Intermediate Timer Events

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

Context

There are a number of business scenarios where 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* (○) and then *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 using one of the following options.
   - Duration:
     - To use expressions, choose *Expression* from the *Duration Based On* dropdown list.

       > ⓘ 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 list. Provide the *Duration* as a numeric value, and choose a *Unit of Time*.

       > ⓘ Note
       >
       > If you are using a static value, make sure that you don't select the *Use Expression* option.

   - Timestamp (absolute time)
     Define the timestamp using a single JUEL expression that must be translated to the absolute time in the ISO 8601 format at runtime. Example: **`${context.maxWaitingTime}`**
5. Choose *Save*.

3.1.3.4.4 Configure Intermediate Escalation Throw Events

Configure an intermediate escalation throw event to communicate the occurrence of an event from a workflow instance to its parent workflow instance in the workflow instances hierarchy.

Context

When workflow execution arrives at an intermediate escalation throw event, an escalation is thrown. This escalation can be caught by a boundary escalation event configured with the same escalation code on a referenced subflow. If there is no matching boundary escalation event, then the flow continues as usual.

In some business scenarios, a workflow might want to communicate the occurrence of a certain event to its parent workflow in the same workflow instances hierarchy. For example, a shipment delayed event can be communicated from a child workflow instance to its parent workflow instance, which can then inform the customer about the delay. The parent workflow instance is the first workflow instance in the hierarchy that catches the escalation event.

Procedure

1. Select *Events* (○) and then *Intermediate Escalation* (◎) and drop it onto the canvas from the palette.
2. Fill in the *Name* and *Documentation* fields for the intermediate escalation event.
3. In the *Intermediate Escalation Event Properties* area, choose the *Details* tab.
4. Provide the escalation code that needs to be propagated to the parent workflow.

> Example
>
> `orderShipped`, `orderDelayed`, `approvalTaskRejected`

> ⓘ Note
>
> JUEL expressions are not supported in escalation codes.

3.1.3.5 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 workflow capability 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 to continue the process.

If multiple sequence flows have a condition that evaluates to true, the first one defined is selected to continue 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.

> ⓘ 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 74].

- **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.

> ⓘ 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 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 e-mail notification can be sent to the employee.

> ⓘ 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 when the exact same steps must be executed in multiple parallel branches.

For more information, see Configure a Parallel Gateway [page 76].

3.1.3.5.1 Configure an Exclusive Gateway

Configure an exclusive gateway in the workflow editor.

Procedure

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

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

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

> ⓘ 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, can you view and configure the *Sequence Flow Properties* . The next step of configuring a condition is only possible for split scenarios.

4. Configure a condition for a sequence flow.

   For more information, see Configure a Sequence Flow [page 75].
5. Choose *Save*.

Related Information

Accelerated Modeling with Speed Buttons [page 82]

3.1.3.5.1.1 Configure a Sequence Flow

Procedure

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

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

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

> ⓘ Note
>
> - You can mark only one outgoing sequence flow as the default.
> - If you want a specific path to execute, for example, only if an employee does not belong to Germany, configure the sequence flow condition as `${context.employee.region!= “Germany”}`. For more information, see Expressions [page 83].

4. Choose *Save*.

3.1.3.5.2 Configure a Parallel Gateway

Configure a parallel gateway in the workflow editor.

Procedure

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

   > ⓘ Note
   >
   > A unique *ID* is generated for every workflow artifact. This ID is read-only.
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 82]

3.1.4 Configure Custom Workflow Attributes

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

Context

At runtime, you can use the respective workflow capability 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 156].

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. **Note** You can only use alphanumeric characters (a-z, A-Z, 0-9) and the underscore character for the ID. The ID must begin with an alphabet character or the underscore but not a number.
Label	A human readable name of the attribute that user interfaces can use to label the attribute.
Type	Data type of the attribute. **Note** Only the data type string is currently supported.
Value	A constant, a JUEL expression, or a mixture of constants and JUEL expressions. It gets resolved when the workflow is created 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, to `${context.aArray[0]}`. A complex object is also not resolved in JUEL expressions. For `${context.complexObject}`, for example, values of that kind of expression always resolve to null. However, it does resolve for a single object element. For example, `${context.complexObject.primitiveProperty}` resolves to the value of the primitive property. `${context}` doesn't contain any nodes. 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

Enable translation for a workflow so that it can be translated.

Prerequisites

- You are using the SAP Business Application Studio to develop applications. See Developing Applications with Workflow Capability [page 21].
- You have created an MTA with a workflow module. See Create a Workflow Module [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 that texts defined for a workflow in the workflow editor are stored in the `i18n_de_DE.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_DE.properties` file in the same folder as the `i18n_de_DE.properties` file.

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

> ⓘ 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
		Custom Attribute Label
Event	Intermediate Message	Name
	Intermediate Timer	
	Intermediate Escalation Event	
Tasks	User Task	Name

Workflow Elements	Translation-Enabled Artifacts	Supported Properties
		Subject
		Description
		Custom Attribute Label
	Service Task	Name
	Script Task	Name
	Mail Task	Name
Gateways	Parallel Gateway	Name
	Exclusive 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 80]

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`.

  > ⓘ Note
  >
  > The name of the folder that the translations reside in and the name of the workflow MUST match. Otherwise, translation does not happen.
- The My Inbox app, Monitor Workflows app, and the workflow capability 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 main 8 languages are considered, and only translations for the supported properties:
  - en_US - English (United States)
  - de_DE - German (Germany)
  - fr_FR - French (France)
  - ja_JP - Japanese (Japan)
  - pt_BR - Portuguese (Brazil)
  - ru_RU - Russian (Russia)
  - es_ES - Spanish (Spain)
  - zh_CN - Chinese (PRC)

3.1.6 Transport Workflows Between Accounts

Use SAP Cloud Transport Management in the SAP BTP, Cloud Foundry environment to transport content from one subaccount to another.

Workflows can be modeled 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 SAP Cloud Transport Management, you can transport all the modules that are part of the MTA project to the target subaccounts in a single transport action. For more information, see What Is Cloud Transport Management.

> ⓘ 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 it in the `mta.yaml` file.
- Enable parallel deployment of application modules in the multitarget application (MTA) deployment descriptor (`mta.yaml`). For more information, see the *Parallel Deployment* section in Order of Deployment and Modules.

Context

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

Procedure

1. Save any workflow resources you've modified.
2. Modify the 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, define it by using the `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 by navigating to *Services* › *Service Instances* in the cockpit.

        Sample Code

        ```
        resources:
          - name: <existing_workflow_instance_name>
            type: org.cloudfoundry.existing-service
        ```
    - If there's no workflow instance created, create a new workflow instance. Define it using the `org.cloudfoundry.managed-service` resource type together with the resource name provided by the user.

        Sample Code

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

For more information about resources types and parameters, see Multitarget Application Structure.

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

   For more information, see the following:

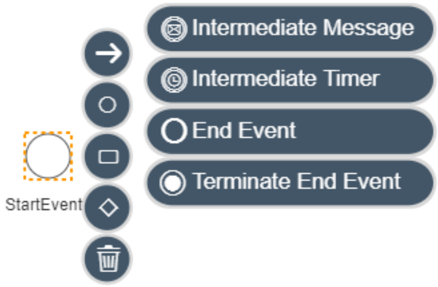
   For SAP Web IDE: Packaging and Deploying Applications to Production Systems

   For SAP Business Application Studio: Build and Deploy.

   > ⓘ 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
○	Allows you to model the following events: • ○ End Event: Creates an end event. • ✉ Intermediate Message: Creates an intermediate message event. • ◎ Intermediate Timer: Creates an intermediate timer event.

Speed Button	Description
▭	Allows you to model the following tasks: - User Task: Creates a user task. - Service Task: Creates a service task. - Script Task: Creates a script task. - Mail Task: Creates a mail task.
◇	Allows you to model the following gateways: - Exclusive Gateway: Creates an exclusive gateway. - Parallel Gateway: Creates a parallel gateway.
→	Creates a sequence flow from one flow element to another. ⓘ 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 highlighted, 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 that 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 54]
Workflow instance information	`${info.workflowInstanceId}`	`$.info.workflowInstanceId`	Getting Information About the Environment in Accessing Contextual Information During Execution of Script Tasks [page 58]
Workflow instance roles	`${roles.adminGroups}`	`$.roles.adminGroups`	Get and Set Instance-Specific Roles [page 60]
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 58]

For a detailed overview of the JUEL expressions and the corresponding script task API, see Cheat Sheet for Workflow Expressions [page 87].

Property Navigation and Text Expressions

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

There are a number of other types of expression:

- 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 75] as follows:
  `${context.employee.region!= “Germany” && context.employee.isManager == true}`
- You must specify the variable to retrieve the request body when calling external services (see Configure Service Tasks [page 46]) as follows:
  `${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 46]) 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 expressions are 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

ⓘ 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
	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 the workflow capability 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 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 54] 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.

Type and Description	Script Task Object/Property Examples (Read/Write Access)	JUEL Expression Examples (Read-only)	More Information
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}`

Type and Description	Script Task Object/Property (Read/Write Access)	JUEL Expression Examples (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}`

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
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.decision`
		`${usertasks.usertask1.last.id}`
		`${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.decision}`
		`${usertasks.usertask1.last.createdAt}`
		`${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

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. For more information, see Open 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 92].
- MTA project containing workflow module and custom task UI. For more information, see MTA Project Containing Workflow Module and Custom Task User Interface [page 93].

- MTA project containing workflow module and SAP Fiori launchpad. For more information, see MTA Project Containing Workflow Module and SAP Fiori Launchpad [page 97].
- 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 102].
- 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 105].

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 91].

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* ◀.

   > ⓘ 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*.

   > ⓘ 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 80].

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 [page 93]
- MTA Project Containing Workflow Module and Custom Task UI Using Workflow Forms [page 97]

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 an 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 91].

Procedure

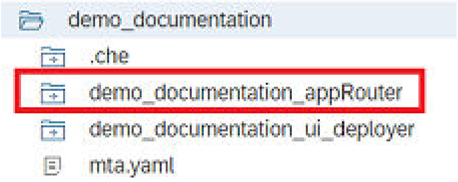
1. Log in to the SAP Web IDE Full-Stack application.
2. From the navigation pane, choose </> (Development) and navigate to the *Workspace* folder.
3. Create a multitarget application (MTA) project in the SAP BTP, Cloud Foundry environment. For more information, see Create a Project from Scratch.

   > ⓘ Note
   >
   > Make sure 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 92].
5. Create an HTML5 module. For more information, see Developing HTML5 Modules.
6. Select *Show hidden files* (👁) to display hidden files.

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

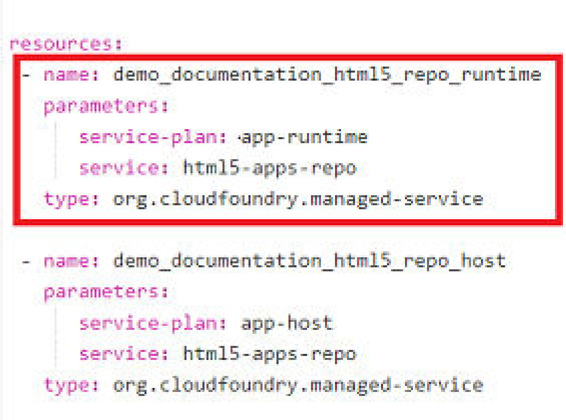
7. Delete the `<MTA ID>_appRouter` module from the MTA project.



8. In the *Code editor*, open the `mta.yaml` file under your MTA project folder.

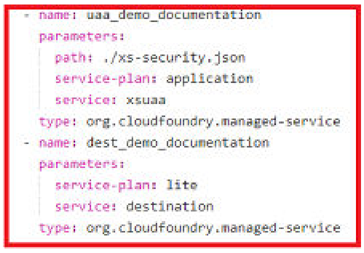
   For the application router module:

   1. Remove `<MTA ID>_html5_repo_runtime` from the `resources` section 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
```

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



```
- name: uaa_demo_documentation
  parameters:
    path: ./xs-security.json
    service-plan: application
    service: xsuaa
  type: org.cloudfoundry.managed-service
- name: dest_demo_documentation
  parameters:
    service-plan: lite
    service: destination
  type: org.cloudfoundry.managed-service
```

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 capability.

> ⓘ Note
>
> The `workflow_rest_url` is the endpoint of the workflow capability 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 SAPUI5 application, proceed with the following steps:
    a. Legacy: Set the Task and Task Context Models [page 125]
    b. Legacy: Bind a UI Element to the Task Context Model [page 123]
    c. Add Task Completion Buttons to My Inbox [page 121]
    d. (Optional) Access the User Task Data [page 115]
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 80].

Results

The page element of the `webapp/view/<view name>.view.xml` should look like this:

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 like this:

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 91].

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
   >
   > Make sure **not** to select *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 92].
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 80].

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 91].

Procedure

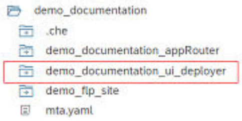
1. Log in to the SAP Web IDE Full-Stack application.
2. From the left pane, choose </> (Development) and navigate to the *Workspace* folder.
3. Create a multitarget application (MTA) project. For more information, see Create a Multitarget Application and Add Modules.

   > ⓘ Note
   >
   > Make sure that you select the *Use HTML5 Application Repository* option 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 92].
5. Select *Show hidden files* (👁) to display hidden files.

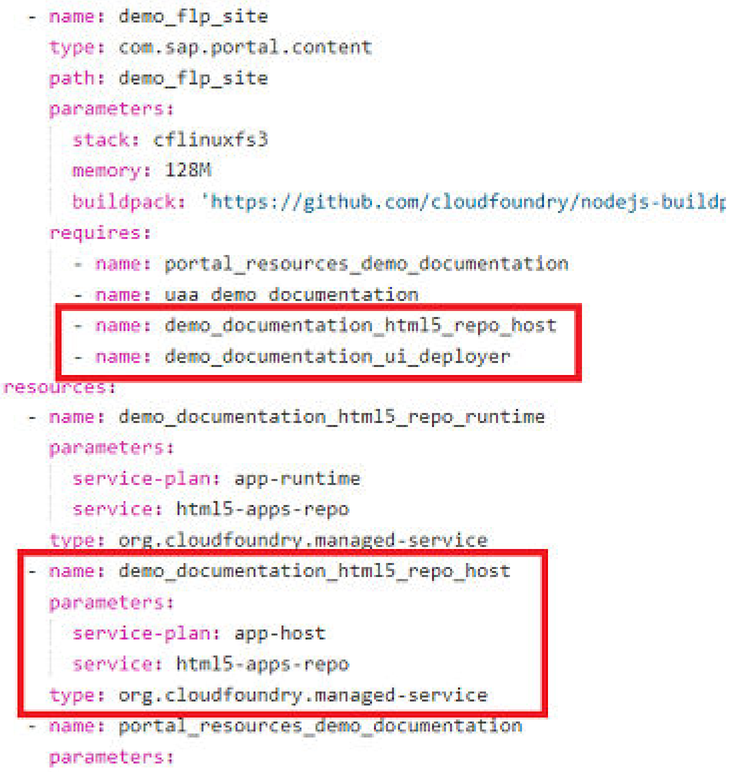
   > ⓘ Note
   >
   > Make sure that you have the application router module and the deployer module present under the MTA project that you created.

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 the 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.

   demo_documentation
   .che
   demo_documentation_appRouter
   demo_documentation_ui_deployer
   demo_flp_site
   mta.yaml

9. In the *Code Editor*, open the `mta.yaml` file located under your MTA project folder.
   - For the 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 a dependency to the workflow capability in the `requires` section of the SAP Fiori launchpad site module.

> ⓘ Note
>
> - Verify that you have provided correct indentations using spaces in the `mta.yaml` file.
> - `<workflow_capability_instance_name>` is the workflow capability 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_capability_instance_name>
    - name: uaa_<MTA ID>
```

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

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

- Add a `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
```

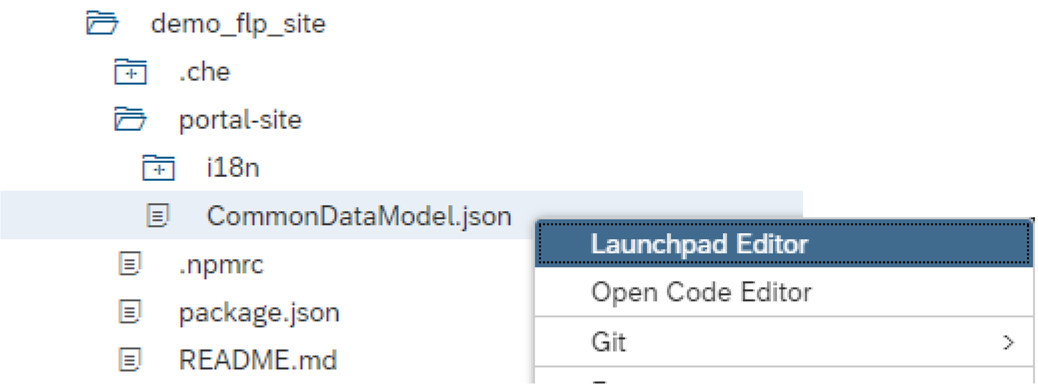
- For the application router module, add the dependency to the workflow capability and the uaa service under the `requires` section of the application router module.

```
- 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_capability_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 80].
13. Add the required roles to users. For more information, see Authorization Configuration [page 258].
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 91].

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
   >
   > Make sure 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 92].
5. Create an HTML5 module. For more information, see Developing HTML5 Modules.
6. Select *Show hidden files* (👁) to display hidden files.

   > ⓘ Note
   >
   > Make sure 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 capability.

   > ⓘ Note
   >
   > The `workflow_rest_url` is the endpoint of the workflow capability 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. Legacy: Set the Task and Task Context Models [page 125]
    b. Legacy: Bind a UI Element to the Task Context Model [page 123]
    c. Add Task Completion Buttons to My Inbox [page 121]
    d. (Optional) Access the User Task Data [page 115]
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 we refer to as the MTA ID in subsequent steps.
11. In the *Code Editor*, open the `mta.yaml` file located under your MTA project folder.
    - For the SAP Fiori launchpad module:
        1. Add a dependency to the workflow capability in the `requires` section of the SAP Fiori launchpad site module.

> ⓘ Note
>
> - Verify that you have provided correct indentations using spaces in the `mta.yaml` file.
> - `<workflow_capability_instance_name>` is the workflow capability 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_capability_instance_name>
    - name: uaa_<MTA ID>
```

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

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

> ⓘ Note
>
> - You have created an HTML5 module in your MTA project, so the `uaa` dependency is already available.
> - The `dest_<MTA ID>` is 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.

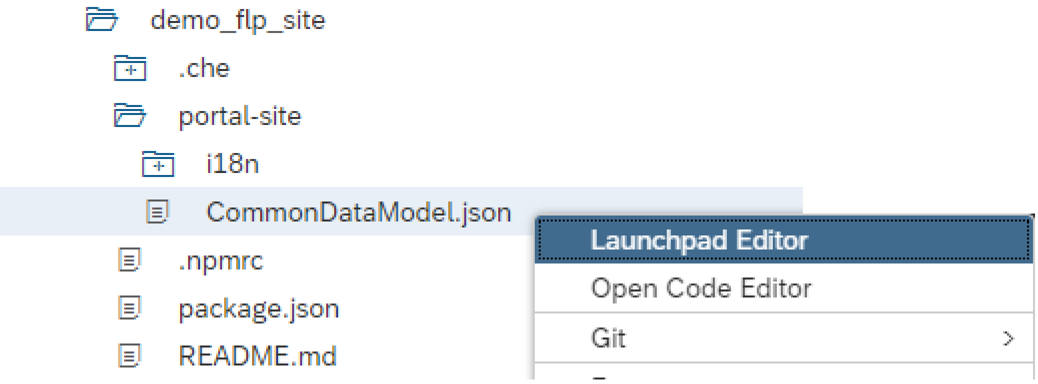
- For the application router module, add a dependency to the workflow capability under the `requires` section of the application router module.

```
- 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_capability_instance_name>
    - name: uaa_<MTA ID>
```

12. In the `xs-app.json` file, inside the `<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 + to add tiles.
    c. Add each tile by choosing + 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 80].
15. Add the required roles to users. For more information, see Authorization Configuration [page 258].
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 in 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 91].

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 a workflow module. For more information, see MTA Project Containing Workflow Module [page 92].
4. Create a Start UI. For more information, see Creating a Custom Start UI [page 126].

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.

> ⓘ Note
>
> Make sure 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 we refer to as the MTA ID in subsequent steps.
9. In the *Code Editor*, open the `mta.yaml` file located under your MTA project folder.

- For the SAP Fiori launchpad module:
    1. Add a dependency to the workflow capability in the `requires` section of the SAP Fiori launchpad site module.

    > ⓘ Note
    >
    > - Verify that you have provided the correct indentations using spaces in the `mta.yaml` file.
    > - `<workflow_capability_instance_name>` is the workflow capability 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_capability_instance_name>
        - name: uaa_<MTA ID>
    ```

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

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

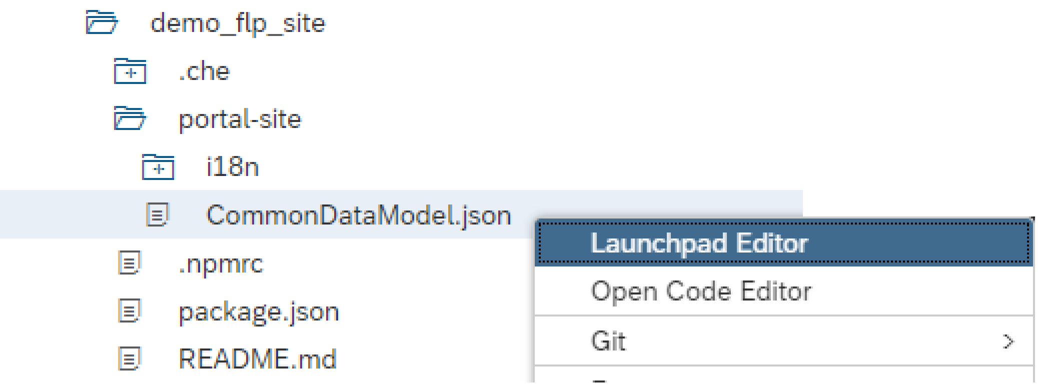
> ⓘ Note
>
> - You have created an HTML5 module in your MTA project, so the `uaa` dependency is already available.
> - The `dest_<MTA ID>` is created in the `resources` section when you add an HTML5 module to your MTA project. You can delete this resource if it is not required.

- For the application router module:
  1. Add a dependency to the workflow capability 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_capability_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 + to add tiles.
    c. Choose + 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 80].

12. Add the required roles to users. For more information, see Authorization Configuration [page 258].
13. Access the SAP Fiori launchpad with the 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 91].

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 capability, define it by using the `org.cloudfoundry.existing-service` resource type along with the existing resource name.
      You can find the existing instance name of the workflow capability by navigating to ▸ *Services* ▸ *Service Instances* ▸ in the cockpit.

      Sample Code

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

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

      Sample Code

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

    For more information about resources types and parameters, see Multitarget Application Structure.

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

   Sample Code

   ```
   requires:
     - name: <workflow_capability_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.

Prerequisites

You have created a multitarget application project and a workflow module. For more information, see Model Workflows in a Multitarget Application Project with SAP Web IDE [page 91].

> Note
>
> Under the workflow module, delete all of the folders except for the `.che` folder.

Procedure

1. If your workflow project contains SAPUI5 content, create separate HTML5 modules and migrate them separately. For more information, see Developing HTML5 Modules.
2. To migrate content other than SAPUI5 content from a workflow project:
   a. In the workflow project, select all folders except for 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. In the *User Task Properties* area, choose the *Details* tab.
   c. In the *Recipients* tab, replace the value of *Users* from ID to e-mail 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*.

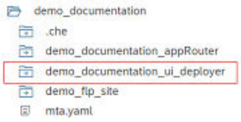
h. Choose *OK*.

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

> ⓘ 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**.

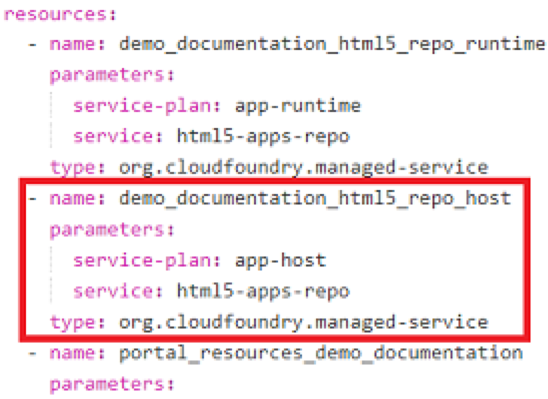
4. If your MTA project doesn't have an HTML5 module, perform the following steps.

a. Delete the `<MTA ID>_ui_deployer` module if the dependency is already available.



```
demo_documentation
  .che
  demo_documentation_appRouter
  demo_documentation_ui_deployer
  demo_flp_site
  mta.yaml
```

b. Open the `mta.yaml` file under your MTA project in the *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. In 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 152].

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 80].

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 the workflow capability, you can create user interfaces (UIs) 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 capability, you can access the workflow capability runtime. On top of these APIs, you can develop scenario-specific user interfaces for user tasks or workflow start UIs.
  - Creating a Custom Start UI [page 126]
    With SAPUI5, this option gives you a high level of control over the UI.
  - Creating a Custom Task UI [page 112]
    This option enables you to create a simple, straightforward UI using predefined elements.
- Creating a Workflow Form [page 136]
  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 156]

3.2.1 Creating a Custom Task UI

You can create a custom task UI and use it as the 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 *SAP Fiori* application type.
  - *Additional SAP Extension Workflow Module*
- You have created a new MTA to contain your custom task UI.

> ⓘ Note
>
> Please check Conventions, Restrictions, and Limits [page 8] for more information about the support of SAPUI5 UIs with internal (hash-based) navigation.

Procedure

1. In SAP Business Application Studio, open your workspace, for example, the *projects* folder the MTA that you just created.
2. Right-click the `mta.yaml` file of your MTA, and choose *Create MTA Module from Template*.
3. Select the *Approuter Configuration* module template, and define the following configuration settings.

Field	Value
Select your HTML5 application runtime	**`Managed Approuter`**
Enter a unique name for the business solution of the project	Enter the business solution name of your application.
Do you plan to add a UI?	**`Yes`**

> ⓘ Note
>
> You can add the approuter configuration only once to a multitarget application.

4. Choose *Next* to complete the configuration.
5. Right-click the `mta.yaml` file of your MTA, and select *Create MTA Module from Template*.
6. Select the *Workflow UI* module template, and define a name for your workflow task UI module. Then choose *Next*.
7. Select the *Task UI* type, and choose *Next*.
8. To customize your task UI module, provide the following configuration settings:

Field	Value
Enter a view name	The name of the UI5 XML view of your generated task UI.
Enter an application title	The title of your task UI application.
Enter an application namespace	The namespace to be considered during the generation of IDs for the SAPUI5 code of your task UI.
Enter a description	The description of your task UI application to be generated in your app's `package.json` file.

9. Choose *Finish* to generate the workflow task UI. This may take a few seconds.

Results

After building and deploying the custom task UI, a task can be approved or rejected using the task completion buttons in My Inbox. For more information about deploying, see Build and Deploy the Workflow Module [page 80].

Workflows in SAP Build Process Automation

Make sure that the destination of the subaccount to which you deploy points to the URL of a service instance of SAP Build Process Automation. In addition, the destination must have valid credentials for the service instance. Typically, this destination is created by running the booster for SAP Build Process Automation. However, you can also create a service instance manually. See Configure SAP Build Process Automation Destinations.

This destination must specify additional properties, for example, these fields:

Field	Value
endpoints	`{"api": https://spa-api-gateway-bpi-eu-prod.cfapps.<region>.hana.ondemand.com}`
sap.cloud.service	`com.sap.spa.processautomation`

When you build your application and if you receive an error message stating that your UI5 CLI installation is outdated, update the `@ui5/cli` version in the `package.json` file of the UI module. You can use version 3 or higher, for example, "^3.4.0".

Related Information

Configure a Custom Task User Interface Using an HTML5 App [page 33]

3.2.1.1 Adapt Routes for Use with Workflow Management Service Instance

All new custom task or start UIs reference the SAP Build Process Automation service instance instead of to the SAP Workflow Management one.

Context

If you want to use the UI for an SAP Workflow Management service instance, you must adapt the `xs-app.json` file because the routes in new task and start UIs point to SAP Build Process Automation routes

Procedure

1. In SAP Business Application Studio, open the `xs-app.json` file that you have generated as described in Creating a Custom Task UI [page 112] or Creating a Custom Start UI [page 126].

   Sample Code

   ```
   {
     ...
     "routes": [
       {
         "source": "^/bpmworkflowruntime/(.*)$",
         "target": "/public/workflow/rest/$1",
         "service": "com.sap.spa.processautomation",
         "endpoint": "api",
         "authenticationType": "xsuaa"
       },
       {
         ...
       }
     ]
   }
   ```

2. Update the routes section of the `xs-app.json` file as follows:

Sample Code

```
{
  ..."
  ,
  "routes": [
    {
      "source": "^/bpmworkflowruntime/(.*)$",
      "target": "/$1",
      "service": "com.sap.bpm.workflow",
      "endpoint": "workflow_rest_url",
      "authenticationType": "xsuaa"
    },
    {
      ...
    }
  ]
}
```

3. Save your changes.

3.2.1.2 Access the User Task Data

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

Context

My Inbox passes the additional data to a custom task UI through the SAPUI5 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`
- `ConfidenceLevel`

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

`ConfidenceLevel` has a value between 0 and 1.

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 Legacy: Set the Task and Task Context Models [page 125].

Access the My Inbox Integration UI API

You can add task completion buttons to 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 of the action buttons, see My Inbox UI Integration API Reference.

Procedure

1. In the *HTML5Module* folder, open ▶ *webapp* ▶ *Component.js* ▶.
2. To enable 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 121].
3. Call the relevant function 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
    ...
}
```

3.2.1.2.1 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`/ `Positive` or `Reject`/ `Negative`)
`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 the successful addition of the button to the footer.

disableAction

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

Parameters

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

Return Value

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

disableAllActions

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

Return Value

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

enableAction

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

Parameters

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

Return Value

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

enableAllActions

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

Return Value

success : A Boolean representing the 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 SAP__`Origin` parameter for the specific task.
`taskInstanceId`	string	Value for the `InstanceId` parameter 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 `SAPOrigin` or `taskInstanceId` parameters are passed with an 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 the successful removal of the button from the footer

setShowFooter

Shows or hides the page footer.

Parameters

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

setShowNavButton

Shows or hides the navigation button in the page header.

Parameters

Name	Type	Description
`showNavButton`	boolean	Flag representing whether to show or hide the navigation button in the pae header. The default value is false.
`navEventHandler?`	function	Function to be called when the 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 SAP__Origin parameter for the specific task.
taskInstanceId	string	Value for the InstanceId parameter 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 SAPOrigin or taskInstanceId parameters are passed with an empty value or if the task list could not be updated (due to network issues).

3.2.1.2.2 Add Task Completion Buttons to My Inbox

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

Prerequisites

The task models are set. See Legacy: Set the Task and Task Context Models [page 125].

Procedure

1. In the *HTML5Module* folder, open ▸ *webapp* ▸ *Component.js* ◂.
2. To enable 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 156]. The actual task outcome can be added to the task context during completion. In the sample code, this is the approved property. The outcome is available for use 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,
        /* TODO: PATCH API merges with existing context, so only specify
the parts
           of the UI context that need an update, for example, ”approved”
*/
        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 Legacy: Set the Task and Task Context Models [page 125]).

   {·} 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 `completeTask` function is called in both actions but with different approval status.
> - If you haven't specified the additional type parameter, the custom action button appears with the default appearance.
> - You can add 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 My Inbox. For more information about deploying, see Build and Deploy the Workflow Module [page 80].

3.2.1.3 Legacy: 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 the `view.xml` file and bind it to an existing text property of the context JSON model.

Prerequisites

You have set the task models, see Legacy: Set the Task and Task Context Models [page 125].

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 like this:

   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 80]), 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, so 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 121].

3.2.1.4 Legacy: 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.

3.2.2 Creating a Custom Start UI

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

Prerequisites

- You have created a dev space in SAP Business Application Studio with the following extensions:
  - *SAP Predefined Extension MTA Tools* that is available, for example, with the SAP Fiori application type.
  - *Additional SAP Extension Workflow Module*
- You have created a new MTA to contain your custom start UI.

Context

In this use case, there is a particular workflow definition deployed into the workflow capability runtime. A user interface is needed which would allow 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 136]). To build a custom start UI, do the following:

Procedure

1. In SAP Business Application Studio, open your workspace, for example, the *projects* folder or your newly created MTA.
2. Right-click the `mta.yaml` file of your MTA, and choose *Create MTA Module from Template*.
3. Select the *Approuter Configuration* module template, and define the following configuration settings. Then choose *Next* to complete the configuration.

Field	Value
Select your HTML5 application runtime	**`Managed Approuter`**
Enter a unique name for the business solution of the project	Enter the business solution name of your application.
Do you plan to add a UI?	**`Yes`**

> Note
>
> You can add the approuter configuration only once to a multitarget application.

4. Right-click the `mta.yaml` file of your MTA, and select *Create MTA Module from Template*.
5. Select the *Workflow UI* module template, and define a name for your workflow start UI module. Then choose *Next*.
6. Select the *Start UI* type, and choose *Next*.
7. To customize your start UI module, provide the following configuration settings:

Field	Value
Enter a view name	The name of the SAPUI5 XML view of your generated start UI.
Enter an application title	The title of your start UI application.
Enter an application namespace	The namespace to be considered during the generation of IDs for the SAPUI5 code of your start UI.
Enter a workflow ID	The ID of the workflow to be started with the start UI.
Enter a description	The description of your start UI application to be generated in your app's `package.json` file.

8. Choose *Finish* to generate the workflow start UI. This might take a few seconds.
9. Optional. To open the start UI in the central Launchpad, specify a proper inbound cross-navigation configuration in the `manifest.json` of your workflow UI.

Sample Code

```
"sap.app": {
    ...,
    "crossNavigation": {
        "inbounds": {
            "<unique key>": {
                "semanticObject": "app",
                "action": "open",
                "title": "My Title",
                "subTitle": "My Subtitle"
            }
        }
    }
    ...
}
```

For more information, see Descriptor for Applications, Components, and Libraries (manifest.json).

Results

After building and deploying the custom start UI, you can configure it as a tile in SAP Fiori launchpad. See Create Custom Start UI Tiles on the Central Launchpad [page 239]. For more information about deployment, see Build and Deploy the Workflow Module [page 80].

Workflows in SAP Build Process Automation

Make sure that the destination of the subaccount to which you deploy points to the URL of a service instance of SAP Build Process Automation. In addition, the destination must have valid credentials for the service instance. Typically, this destination is created by running the booster for SAP Build Process Automation. However, you can also create a service instance manually. See Configure SAP Build Process Automation Destinations.

This destination must specify additional properties, for example, these fields:

Field	Value
endpoints	`{"api": https://spa-api-gateway-bpi-eu-prod.cfapps.<region>.hana.ondemand.com}`
sap.cloud.service	`com.sap.spa.processautomation`

When you build your application and if you receive an error message stating that your UI5 CLI installation is outdated, update the `@ui5/cli` version in the `package.json` file of the UI module. You can use version 3 or higher, for example, "^3.4.0".

3.2.2.1 Adapt Routes for Use with Workflow Management Service Instance

All new custom task or start UIs reference the SAP Build Process Automation service instance instead of to the SAP Workflow Management one.

Context

If you want to use the UI for an SAP Workflow Management service instance, you must adapt the `xs-app.json` file because the routes in new task and start UIs point to SAP Build Process Automation routes

Procedure

1. In SAP Business Application Studio, open the `xs-app.json` file that you have generated as described in Creating a Custom Task UI [page 112] or Creating a Custom Start UI [page 126].

   Sample Code

   ```
   {
     ...
     "routes": [
       {
         "source": "^/bpmworkflowruntime/(.*)$",
         "target": "/public/workflow/rest/$1",
         "service": "com.sap.spa.processautomation",
         "endpoint": "api",
         "authenticationType": "xsuaa"
       },
       {
         ...
       }
     ]
   }
   ```

2. Update the routes section of the `xs-app.json` file as follows:

   Sample Code

   ```
   {
     ..."
     ,
     "routes": [
       {
         "source": "^/bpmworkflowruntime/(.*)$",
         "target": "/$1",
         "service": "com.sap.bpm.workflow",
         "endpoint": "workflow_rest_url",
         "authenticationType": "xsuaa"
       },
       {
         ...
       }
     ]
   }
   ```

3. Save your changes.

3.2.2.2 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 runtime.

Context

In this use case, a particular workflow definition is deployed into the workflow runtime. A user interface is needed to allow end users to start the instances of the corresponding workflow. In addition, the users must be able to specify 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 136]). To build a custom start UI, do the following:

Procedure

1. Create an HTML5 Module for the Start UI [page 130].
2. Define the Route [page 131]
3. Extend the View [page 131]
4. Extend the Controller [page 132]

Related Information

SAP Web IDE Full-Stack Documentation

3.2.2.2.1 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, 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.

3.2.2.2.2 Define the Route

Define the route for the workflow capability in the application configuration file.

Procedure

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

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"
               }]
}
```

3.2.2.2.3 Extend the View

Context

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

Procedure

In the view XML file created in the *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>
```

3.2.2.2.4 Extend the Controller

Procedure

In the `controller JS` file created in the *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 133]
- Bind an Action to the Button Push Event [page 135]
- Implement Workflow Instance Start [page 134]
- Implement XSRF Token Fetch [page 134]

Results

The extension parameter of the controller now looks like this:

ⓘ 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;
    }
}
```

3.2.2.2.4.1 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: ""
    }));
}
```

3.2.2.2.4.2 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;
}
```

3.2.2.2.4.3 Implement Workflow Instance Start

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

In this example, the input of the user is used in the context of the workflow instance; namely, in its text field. For more information, see Using Workflow APIs [page 156]. In addition, the response of the call is assigned to the corresponding property in the data model.

ⓘ 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));
        }
    });
}
```

Note

Substitute the `<your workflow ID>` part of the URL with the ID of the deployed workflow definition.

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

3.2.2.2.4.4 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.

> ⓘ Note
>
> If you are using the free service plan for SAP Build Process Automation, you can only deploy up to 500 legacy workflow forms.

- Start Forms
  Initiate a workflow based on form input. To provide a start form for your end users, integrate it into your SAP Fiori launchpad. See Configure a Start-Form-Based Workflow Start App [page 215].
- 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 215]. 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 tasks 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.

> ⓘ 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*.

   > 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.

> 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 a Form

You can build your form by using fields or collections. You can arrange it with sections and subsections.

Related Information

Add Fields [page 139]
Add Collections [page 144]

3.2.3.1.1.1 Add Fields

Build forms using fields that you can arrange using sections and subsections.

Procedure

1. Double-click a file to open it.
2. Add a field to your form by choosing *Add Field* at the top of the field table.

   Where the field appears depends on whether you've selected an existing field, any sections, subsections, or collections.

   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 147] and Add Collections [page 144].
3. In 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 ID.

   > ⓘ 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 the fields in your start form don’t show initial values. A workflow start creates the bound properties in the empty workflow context. For more information, see Automatic Model Initialization [page 150].

Task Form

Set the mode of your fieldWhen 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 [page 142]), 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 150].

The syntax follows the JUEL style described in Expressions [page 83]. Fields on a 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 144].

Context Path Suggestions

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 setThe 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.

> ⓘ Note
>
> You can only access the context model 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 tothe `orderDateTime` property within the first item of the invoices array.

For an example of collection fields, see Add Collections [page 144].

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.

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 the 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

> ⓘ 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.

> ⓘ 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 137]).
>
> 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 only set the display-only mode if adding and deleting items to the collection itself is disabled. See Add Collections [page 144].

8. Set the constraints of your field.

   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 149]).

> ⓘ 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* tab 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.

   The type-specific validations apply for both (see Form Validation [page 154]).

   - To add an entry, choose *Add* from the toolbar menu. You can add up to 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.

> ⓘ 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 subsections, collections themselves are bound to a context path, namely an array within the collection, which specifies the rows of the table.

Procedure

1. Double-click a file to open it.
2. Add a collection to your form by choosing *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 147].
3. (Optional) In the *Properties* view, name the collection by entering text for the collection title, which is rendered as the table title in the form.
4. Enter the *ID* of the collection or use the automatically generated one.

   > ⓘ 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 a 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 83].

> **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
>     }]
>   }
> }
> ```
>
> If 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 only access the context model using dot notation. Conditions and literals aren’t supported. Make sure that you use a valid path to a property in the context.

6. For task forms, define the interaction configuration of the collection.

Allow Adding and Deleting Items	Result
Enabled/Checked	The end user can add new items to the rendered table in the task form. The end user can delete existing and newly added items.
Disabled/Unchecked	The end user can't change the number of items in the rendered table. That means that the end user can't add new items or delete existing ones.

> ⓘ Note
>
> You can only enable adding and deleting items in task forms if there aren't any fields with display-only mode in the collection. To enable adding and deleting items, set the mode to editable for all fields of the collection.
>
> For rendered tables in start forms, the end user can always add new items and delete existing and new items by default. The interaction configuration isn't visible in the start form editor.
>
> The interaction configuration affects only the rendered form and not the workflow runtime itself. The end user can still modify a collection and its items using the REST API or script tasks.

7. 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 click *Add Field*. For more information, see Add Fields [page 139]. 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. 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.

   > ⓘ Note
   >
   > To deselect a selected field, click it again.

2. Set the mode of your form.

   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 cannot modify the value on the UI. The fields are in display mode.

   A form-wide display-only mode overwrites individual display-only field modes. You can't change the mode for individual fields. However, your previously modeled modes, as well as constraints on the individual fields, are preserved if you switch to form-wide *Editable* mode again.

   > ⓘ 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. If your form already has sections, a new section is added.

If you need a more granular grouping, you can select a section and choose *Add Subsection*. If the selected section already contains fields or collections, they are moved into the new subsection.

> ⓘ 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 another section, then it’s converted into a subsection. The latter applies only to sections that contain fields but not to sections that contain subsections. By using copy and paste, you can also move elements from one form to another.

You cannot paste collections or fields that are part of a collection into a start form.

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 150].

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.

    > ⓘ 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

> ⓘ 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 83]. 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 75]

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.

> ⓘ 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 |

> ⓘ 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.
>
> Because 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 139]. This is what it looks like:

Workflow (Task) Context	Binding	UI Model (Initial)	UI Model with Data

{}	${context.myNode1.myNode2.myProperty}	`{`
		`   myNode1: {`
		`      myNode2:{}`
		`   }`
		`}`
			`{`
			`   myNode1: {`
			`      myNode2: {`
			`myProperty:`
			`"anyValue"`
			`      }`
			`   }`
			`}`

Task forms only. Missing context properties are also created for an existing element in an 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 could 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 different 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 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 137].

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 Changed)
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 139]

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.

> ⓘ 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

The workflow capability has 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 capability maintains the relation between a workflow instance, its files, and additional metadata. You can choose whether end users are allowed to see attachments in a form 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 165].

Attachments are only available for task forms. As the feature just supports "read-only", no attachments are rendered in a start form.

Take the following into consideration:

- Missing Repository Connectivity
  The workflow capability 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 such as name, creator, or size might not be shown. Furthermore, attachments cannot be downloaded.
- Missing Attachments in a Form
  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 attachments is configured, see Configure Document Management for Workflow Capability Attachments [page 206].

Procedure

1. In the fields table of your task form, make sure that no field is selected.

   > ⓘ 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 the workflow capability.

The workflow capability 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 capability.

For information about who can execute these actions, see Authorization Configuration [page 258].

Access the API Documentation

See the workflow capability documentation on the SAP Business Accelerator Hub.

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 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 managed transparently by the application router for SAP BTP, Cloud Foundry environment applications that are bound to the workflow capability.
  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 261].

> ⓘ Note
>
> For all OAuth2-based workflow APIs, you don't 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`.

Translation

Certain APIs evaluate the `Accept-Language` HTTP header to determine the language in which fields that support translation are sent. For more information about how to provide translated texts for workflow content, see Translate Workflows [page 78].

> ⓘ Note
>
> Workflow APIs do not support translation for any fields other than those mentioned above. Especially, error messages for unsuccessful requests are always provided in English. To look up scenario-specific translations in the user interface, use the `code` fields of the error information structure that is returned in the response as a technical identifier.

Libraries and Integration

The SAP Cloud SDK provides pregenerated client libraries to consume the API of the workflow capability in Java, JavaScript, and TypeScript applications on SAP Business Technology Platform (SAP BTP).

It allows convenient invocation of the service in the programming language of your choice, while facilitating connectivity and authentication handling.

For more information, see SAP Cloud SDK for Java and SAP Cloud SDK for JavaScript/TypeScript .

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 capability 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 capability.
      For more information, see Create Service Keys Using the Cockpit in the SAP Business Technology Platform (SAP BTP) documentation.
      See the table for details about the configuration parameters.
    - If you work with a service binding, you find all the configuration parameters that are relevant for the workflow capability 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 about 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`**

Parameter Name	Description	Example
`endpoints.workflow_rest_url` and `endpoints.workflow_odata_url`	URL of the workflow capability	**`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 the workflow capability 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 the workflow capability 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 194].
- You've noted down the following parameters from the procedure Determine Service Configuration Parameters [page 158]: `clientid`, `clientsecret`, `url`, `workflow_rest_url`, or `workflow_odata_url`.
- Assign the necessary role collections of the workflow capability API that you want to call, to the user on whose behalf the call to the workflow capability 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 258].

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.

> ⓘ 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` endpoint of the authorization server, for example: `<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 capability API (on behalf of the 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` endpoint of the authorization server, for example: `<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).

> ⓘ 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 asking the user for consent again. 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 way to request a new refresh token.

3. Perform the call to the workflow capability API by sending the access token as the header. Use the endpoints below the base URL from the `workflow_rest_url or workflow_odata_url` service configuration parameter.
   - 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 the workflow capability APIs using OAuth 2.0 authentication.

Prerequisites

- You've created a service instance of the workflow capability with the necessary authorization to invoke the REST APIs as described in Technical Authentication [page 261].
- 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 194].
- You've noted down the following parameters from the Determine Service Configuration Parameters [page 158] procedure: `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 the workflow capability 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 are the values that you noted down as a 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 capability API by sending the access token as the header. Use the endpoints below the base URL from the `workflow_rest_url` or `workflow_odata_url` service configuration parameter.
    - 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 capability is available from the `endpoints.workflow_rest_url` and `endpoints.workflow_odata_url` configuration parameters of the service key, or of the service binding, depending on your application type.

Context

For information about how to access this information, see Determine Service Configuration Parameters [page 158].

> ⓘ Note
>
> If you access the workflow capability 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 you implement, for example, the Application Router, instead of directly referring to the URL defined from the above, you have to refer to application routes and destinations.

Apply the following conventions to the base URL, where necessary:

Procedure

1. For both `endpoints.workflow_rest_url` and `endpoints.workflow_odata_url`, extend the URL with the version of the API. This means that you need to extend the URL with `/v1` as currently both API types only provide a single version.
2. For the Inbox API of type OData, further extend `endpoints.workflow_odata_url` with `/tcm`. For UI migration scenarios, you might be advised by SAP support to use `/tcm/custom` instead.

3.3.5 Modifying the Context of a Workflow Instance

You can modify the context of a workflow instance in RUNNING, ERRONEOUS, or SUSPENDED status.

> 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 ensuring 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 execution of the workflow instance. See the section about suspending or resuming workflow instance.
>   For more information, see /v1/workflow-instances/{workflowInstanceId}.

Override Context

Overriding the 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 overriding (equals the payload of the override operation):
>
> Sample Code
>
> ```
> {
>     variableOverriden: "hello!",
>     variableNestedObject: {
>             variableNested: false,
>             variableNestedNew: "new value"
>         },
> ```

```
    variableNew: "I'm new"
}
```

Patch Context

Patching the context of the workflow instance merges the contents of the context before overriding 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 to the workflow instance context with the corresponding value.

ⓘ Note

Merging happens at all levels of complex object 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

The workflow capability provides APIs to manage relationships 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 capability attachments is configured, see Configure Document Management for Workflow Capability Attachments [page 206].
- 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" }
    ]
   }
  }
 }
}
```

ⓘ 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 Capability Attachments [page 206].

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 the 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" }
  ]
  }
 }
}
```

ⓘ Note

Additional information about an attachment, such as name, creation/modification dates, or 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 a Task Instance

A more common use case might be reading this information while operating on a task instance. The API looks like this:

```
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" }
     ]
   }
  }
}
```

> Note
>
> Additional information about an attachment, such as the name, creation/modification dates, or 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 attachment 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 back end, remain untouched.

This means, that it is not possible to remove a group on task completion, , but you can empty it.

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 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.

Note

The following applies similarly to the APIs of SAP Build Process Automation. The `Workflow Participant Role` corresponds to the `ProcessAutomationParticipant` role and the `Workflow Administrator` role to the `ProcessAutomationAdmin` role in SAP Build Process Automation.

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>",
                "userInterfaceUri": "<New user interface uri>"
 }
```

Where <New subject>, <New description>, <New due date>, <New priority>, <New processor>, <New recipient users>, <New recipient groups>, and <New user interface uri> refer to the values of the task subject, description, due date, priority, processor, recipient users, recipient groups, and user interface URI 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 capability 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 83] 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 258]

Intelligence Within a Workflow – Confidence Levels for Task Instances [page 171]

3.3.8 Intelligence Within a Workflow – Confidence Levels for Task Instances

The confidence level represents the probability of task approval.

It allows you to bring your own artificial intelligence (AI) model to enhance your workflows with an intelligent decision support. If you're looking for a simpler solution, see Configure a Workflow Intelligence Scenario.

These confidence levels can be displayed in custom task UIs to simplify the decision a user has to take. It's especially helpful with repetitive tasks and as a decision support for substitutes. For information about how to configure a custom task UI, see Creating a Custom Task UI [page 112].

To update a task instance with a confidence level, send an HTTP request with the PATCH method to the corresponding API endpoint. Use, for example, the following payload:

Sample Code

```
{
 "confidenceLevel": 0.8
}
```

Here `confidenceLevel` can be any number between `0.0` and `1.0`.

To delete the confidence level for a task instance, provide the value `-1.0`.

For more information, see task patch API.

3.3.9 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.

Log Entry/Event	Description
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.
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.
INTERMEDIATE_TIMER_EVENT_FAILED	Intermediate timer event FAILED in a workflow instance.
CANCELING_BOUNDARY_TIMER_EVENT_TRIGGERED	Boundary timer event triggered the cancellation of the attached user task or referenced subflow 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 or referenced subflow .

Log Entry/Event	Description
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.
EMBEDDED_SUBFLOW_STARTED	Embedded subflow started.
EMBEDDED_SUBFLOW_COMPLETED	Embedded subflow completed.
REFERENCED_SUBFLOW_STARTED	Referenced subflow started.
REFERENCED_SUBFLOW_COMPLETED	Referenced subflow completed.
REFERENCED_SUBFLOW_FAILED	Referenced subflow failed.
REFERENCED_SUBFLOW_CANCELED_BY_BOUNDARY_EVENT	Referenced subflow canceled by a boundary timer event or boundary escalation event.
INTERMEDIATE_ESCALATION_EVENT_EMITTED	Intermediate escalation event thrown for a workflow instance.
CANCELING_BOUNDARY_ESCALATION_EVENT_TRIGGERED	Boundary escalation event triggered the cancellation of the attached referenced subflow and continued the alternative flow.
NONCANCELING_BOUNDARY_ESCALATION_EVENT_TRIGGERED	Boundary escalation event triggered the alternative flow attached to it without canceling the attached referenced subflow.

3.3.10 Error Codes

If an error occurs while working with the workflow capability 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 278]

Error Code/Error Code Prefix	Description	Related Links
`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 83]
`bpm.workflowruntime.destination`	There was a problem resolving or accessing a destination.	Destinations [page 262] Configure the Workflow Capability Mail Destination [page 199]
`bpm.workflowruntime.servicetask`	There was a problem executing a service task.	Configure Service Tasks [page 46]
`bpm.workflowruntime.mailtask`	There was a problem executing a mail task.	Configure Mail Tasks [page 62]
`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 62]
`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 62]
`bpm.workflowruntime.scripttask`	There was a problem executing a script task.	Configure Script Tasks [page 52]
`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 156]

Related Information

SAP Business Accelerator Hub Workflow Capability

3.4 Transition to SAP Build Process Automation

Transition the workflows that you previously used with a service instance of workflow capability to a service instance of SAP Build Process Automation.

SAP Build Process Automation offers many new capabilities and you can easily move your existing workflow definitions to SAP Build Process Automation at your own pace and without loosing any data or investments.

Processes created in the process builder can use all these functionalities, while all existing workflow definitions and workflow instances stay available as described below. You can start to build your first processes in the process builder and reuse existing workflows as subprocesses. See Add a Subprocess to a Process. This enables a stepwise approach for moving to SAP Build Process Automation.

> ⓘ Note
>
> Important: All of this only applies if SAP Build Process Automation is used in the same consumer account. This excludes customers that are in a data center where SAP Build Process Automation is not available. For the bigger picture, see SAP Workflow Management – The Path Forward.

How Existing Workflows Fit In

If you're already using SAP Workflow Management in a consumer account and now start using SAP Build Process Automation in the same consumer account, you can continue using existing workflows without any transition:

- Existing workflow instances are automatically available in the monitoring tools of SAP Build Process Automation.
  Existing workflow definitions also remain available. You can still access these workflow instances and definitions using an API.
- Tasks are available in the local My Inbox of SAP Task Center if you configured it as described in Configure the SAP Build Process Automation Subaccount for SAP Task Center.
- Processes that you create in the process builder of SAP Build Process Automation can invoke existing workflows as subprocesses. See Add a Subprocess to a Process.

We recommend that you always consider rebuilding processes, that is, create workflows as processes in the process builder to leverage the full power of SAP Build Process Automation. However, the workflow editor in SAP Business Application Studio remains available in SAP Build Process Automation and all existing workflows continue to be editable in the workflow editor. You can either continue to deploy workflow definitions form SAP Business Application Studio to a service instance of the workflow capability as long as such an instance exists or you can deploy to a service instance of SAP Build Process Automation. The latter requires minor changes if workflow definitions exist already as described in the following sections. A redeployment is not required if there are no logical changes. Deployed workflow definitions remain available in the runtime. New workflows that were created with the workflow editor are already adapted so that they can be deployed to a service instance of SAP Build Process Automation without the changes described in the following sections.

Service tasks within your existing workflows might use APIs of other capabilities within SAP Build Process Automation. In some cases, the called endpoint might have changed so that the used destination or the service task within the workflow definition requires adjustments. This requires a case-by-case check.

If you have developed custom applications based on workflow capability APIs, then adjust them to use the APIs of SAP Build Process Automation instead.

Things to consider

- If at the time of deletion of your last workflow capability instance, you already have an SAP Build Process Automation instance, the data stays available for use in SAP Build Process Automation.
- To ensure that principal propagation continues to work when transitioning to SAP Build Process Automation, a change needs to be made to the `bpmworkflowruntimeoauth` destination before deleting

the last instance of the workflow capability. See Adapt the bpmworkflowruntimeoauth Destination [page 177].

> ⚠ Caution
>
> It is essential that you create the service instance and adapt the destination of SAP Build Process Automation before you delete the last remaining service instance of the workflow capability across all the spaces in your organization. If you fail to do so, all your data within the workflow capability is irrevocably erased.

To modify your existing workflows so that they are deployed to a service instance of SAP Build Process Automation, do the following:

1. Modify an Existing Workflow Module [page 178]
2. Modify an Existing Workflow [page 180]
3. Modify Existing Start and Task UIs [page 180]

Related Information

Blog: Transition your SAP Workflow Management Solutions to SAP Build Process Automation

3.4.1 Adapt the `bpmworkflowruntimeoauth` Destination

To ensure that principal propagation continues to work when transitioning to SAP Build Process Automation, a change needs to be made to the `bpmworkflowruntimeoauth` destination before deleting the last instance of the workflow capability.

Context

If you already run a scenario based on principal propagation, then you have created a destination named **bpmworkflowruntimeoauth** as described in Create an OAuth Destination [page 198].

Before you performe the transition to SAP Build Process Automation, that is, before **you delete the service instances of the workflow service**, you must retrieve and **specially protected** the credentials with a parameter. This ensures that the credentials are still active even after you have deleted the service. The credentials are necessary to continue using principal propagation.

Procedure

1. Identify the existing service instance of the workflow capability that is already used to provide the configuration in the `bpmworkflowruntimeoauth` destination. The *User* property of the destination contains the matching ID of the service instance.

2. Create a new service key for the identified service instance. To do so, use one of the following options.
    - In the SAP BTP cockpit, use `{"xsuaa": {"credential-type": "binding-secret"}, "keepOauthSecretValidBeyondDeletion": true}` as parameter for the service key.
    - In the SAP BTP Command Line Interface, enter:
      ```
      cf create-service-key <service-instance-name> <any-service-key-name> -c '{"xsuaa": {"credential-type": "binding-secret"}, "keepOauthSecretValidBeyondDeletion": true}'
      ```
3. From the service key, copy the values of the `clientid` and `clientsecret` properties.
4. Locate the `bpmworklowruntimeoauth` destination.
5. While the *User* field already contains the `clientid` value, you must update the *Password* field with the `clientsceret` property from the previous step.
6. Save the destination.

3.4.2 Modify an Existing Workflow Module

Transition an existing workflow module, that was previously used with a service instance of workflow capability, to a service instance of SAP Build Process Automation.

Prerequisites

- You have created a dev space with the following extensions:
    - The SAP Predefined Extension *MTA Tools* that comes, for example, with the *SAP Fiori* or *SAP Cloud Business Application* application types.
    - The SAP Predefined Extension *Workflow Module*.

  For more information, see Create a Dev Space [page 20].
- You have imported an existing multitarget application (MTA) that contains your workflow modules.

Context

Existing workflow modules might still reference the workflow capability service instance. To deploy the MTA to the service instance of SAP Build Process Automation, you must modify the `mta.yaml` file.

> ⚠ Caution
>
> Removing the last service instance automatically deletes this data.
>
> However, if at the time of deletion an SAP Build Process Automation instance exists, the data will not be deleted and stays available for use in SAP Build Process Automation.

Procedure

1. In SAP Business Application Studio, use a text editor to open the `mta.yaml` file that you generated as described in Create a Workflow Module.

   It looks similar to this sample:

   Sample Code

   ```
   modules:
   - name: workflow-module
     type: com.sap.application.content
     path: workflow-module
     requires:
     - name: workflow
       parameters:
         content-target: true
   resources:
   - name: workflow
     type: org.cloudfoundry.managed-service
     parameters:
       service: workflow
       service-plan: standard
   ```

2. Update the `requires` and `resources` sections of the `mta.yaml` file as follows:

   Sample Code

   ```
   requires:
     - name: sap_processautomation
       parameters:
         content-target: true
         service-key:
           config:
             deployUsageScenario: workflow
           name: spa-workflow-service-key
   resources:
   - name: sap_processautomation
     type: org.cloudfoundry.managed-service
     parameters:
       service: process-automation-service
       service-plan: standard
   ```

3. Save your changes.
4. Continue the transition with Modify an Existing Workflow [page 180].

3.4.3 Modify an Existing Workflow

Transition an existing workflow, that was previously used with a service instance of workflow capability, to a service instance of SAP Build Process Automation.

Prerequisites

You've moved the workflow module. See Modify an Existing Workflow Module [page 178].

Context

Service tasks and mail tasks of existing workflows might miss some configuration settings. Transition these workflows, so they can be executed within a service instance of SAP Build Process Automation.

Procedure

1. Right-click the workflow, and open it in the workflow editor.
2. Make any change, for example, by changing the layout of the workflow.
3. If there are custom start and task UIs used in the workflow, then follow the steps in Modify Existing Start and Task UIs [page 180].
4. If your workflow contains any service tasks that access Workflow or Business Rules APIs, adapt the task's destination and paths to the corresponding SAP Build Process Automation APIs. See SAP Build Process Automation API.
5. Save your changes.
6. Build and redeploy your workflow.

3.4.4 Modify Existing Start and Task UIs

Transition existing workflow UI modules (start and task UIs), that were previously used with the service instance of the workflow capability, to the service instance of SAP Build Process Automation.

Prerequisites

- You've created a dev space with the following extensions:
    - The SAP Predefined Extension *MTA Tools* that comes, for example, with the *SAP Fiori* or *SAP Cloud Business Application* application types.

    - The SAP Predefined Extension *Workflow Module*.

  For more information, see Create a Dev Space [page 20].
- You've imported an existing multitarget application (MTA) that contains your workflow modules.

Context

Existing workflow UI modules might reference the service routes of the workflow capability. To deploy the MTA to the SAP Build Process Automation service instance, you must modify the `xs-app.json` file.

Procedure

1. In SAP Business Application Studio, open the `xs-app.json` file that you generated as described in Creating User Interfaces.

   The `routes` configuration looks similar to this sample:

   Sample Code

   ```
   {
     ..."
     ,
     "routes": [
       {
         "source": "^/bpmworkflowruntime/(.*)$",
         "target": "/$1",
         "service": "com.sap.bpm.workflow",
         "endpoint": "workflow_rest_url",
         "authenticationType": "xsuaa"
       },
       {
         ...
       }
     ]
   }
   ```

2. Update the `routes` configuration as follows:

   Sample Code

   ```
   {
     ...
     "routes": [
       {
         "source": "^/bpmworkflowruntime/(.*)$",
         "target": "/public/workflow/rest/$1",
         "service": "com.sap.spa.processautomation",
         "endpoint": "api",
         "authenticationType": "xsuaa"
       },
       {
         ...
       }
     ]
   ```

```
}
```

3. Save your changes.
4. Rebuild and redeploy your MTA.

3.4.5 Modify Existing Task Instances

Transition existing task instances that were previously used with a service instance of workflow capability, to a service instance of SAP Build Process Automation.

Prerequisites

Make sure that the context variables are already set when you update the task instances. Otherwise, they resolve to empty strings.

Context

You can modify the user interface of running task instances. This may be necessary if you switch from a Standalone App Router to a Managed App Router or if you simply want to reference another user interface.

To make these changes, use the GET and PATCH APIs for task instances.

Note that the updated `userInterfaceUri` fully replaces the old version. To dynamically set parts of the URI during updates, you can use Expressions [page 83].

> ⓘ Note
>
> The new `userInterfaceUri` must be a valid URI. Any special characters must be properly URL-encoded.

Supported `userInterfaceUri` values are, for example:

- `sapui5://com.sap.bpm.bpi.forms?formId=12a8b0e1-41f1-4bbc-a41c-5cab6bd16625&data=data%20with%20spaces`
- `sapui5://comsapbpmworkflow.comsapbpmwusformplayer/com.sap.bpm.wus.form.player?formId=form91872d18&formRevision=v1`
- `${context.path}?formId=form91872d18&formRevision=v1`

See the example approach for updating the `userInterfaceUri` of existing task instances.

Procedure

1. Use the GET Task Instances APIs to retrieve the current values of the user interface URI for your task instances.

   For example, query for tasks that have the same `userInterfaceUri` using the `definitionId` parameter.
2. Define new values for the `userInterfaceUri` URIs of your task instances.
3. Use one of the PATCH APIs to update instances individually or in batch. When using the batch API, patching is done asynchronously. Check the job status to ensure that there are no errors.
4. Use the GET APIs to confirm that all task instances have been updated.

   > ⓘ Note
   >
   > When using the `userInterfaceUri` filter in a GET request, you may need to URL-encode the value, particularly if it already includes URL-encoded characters. This is because the filter value is sent as a URL parameter.
   >
   > Example:
   >
   > User Interface: `sapui5://com.sap.bpm.bpi.forms?formId=12a8b0e1-41f1-4bbc-a41c-5cab6bd16625&data=data%20with%20spaces`
   >
   > URL-Encoded User Interface: `sapui5%3A%2F%2Fcom.sap.bpm.bpi.forms%3FformId%3D12a8b0e1-41f1-4bbc-a41c-5cab6bd16625%26data%3Ddata%2520with%2520spaces%C2%A0`

Related Information

Updating Task Properties [page 168]

3.4.6 Expose the Service Instance of SAP Build Process Automation Using Destinations

Make sure that the destination of the subaccount to which you deploy points to the URL of a service instance of SAP Build Process Automation.

Prerequisites

The destination must have valid credentials for the service instance. Typically, this destination is created by running the booster for SAP Build Process Automation. However, you can also create a service instance manually. See Configure SAP Build Process Automation Destinations.

Procedure

1. Specify additional properties for the destination, for example, these fields:

Field	Value
endpoints	`{"api": https://spa-api-gateway-bpi-<region_without_number>-prod.cfapps.<region>.hana.ondemand.com}`
sap.cloud.service	`com.sap.spa.processautomation`

2. If you receive an error message when building your application that states that your UI5 CLI installation is outdated, update the `@ui5/cli` version in the `package.json` file of the UI module. You can use version 3 or higher, for example, "^3.4.0".

3.4.7 Migrating Neo Workflows

To migrate your workflows from the Neo environment to SAP Build Process Automation, the following prerequisites must be met.

Prerequisites

- You have been subscribed to SAP Build Process Automation See Initial Setup in the SAP Build Process Automation documentation.
- Required SAP Build Process Automation roles have been assigned. See Authorizations in the SAP Build Process Automation documentation.
- The Cloud Foundry organization has been enabled in the same subaccount and a service instance of SAP Build Process Automation has been created in your Cloud Foundry space.
- You've assigned the space developer role to your user in the Cloud Foundry space. See Add Space Members Using the Cockpit.
- SAP Build Work Zone, standard edition has been subscribed in the subaccount of the above Cloud Foundry space.
- Optional: SAP Build Work Zone has been configured so that SAP Build Process Automation UIs are available. See Configure SAP Build Work Zone for SAP Build Process Automation in the SAP Build Process Automation documentation.
- Destinations used in the workflows for service tasks and mail tasks have been created in the subaccount. See Destinations on SAP Help Portal.
- SAP Business Application Studio has been subscribed. We recommend that you subscribe to SAP Business Application Studio in the same subaccount that you use for SAP Build Process Automation. See Subscribe to SAP Business Application Studio.
- The SAP BTP, Neo environment subaccount with SAP Web IDE Full-Stack is enabled.
- You've exported your workflow including SAPUI5-based task UIs from SAP Web IDE Full-Stack and downloaded the zip files to your local hard disc.

Restrictions

The following restrictions apply for migrating your Neo workflows:

- There’s no SAP Task Center integration on Neo. That is, there's no inbox provided that federates task instances from the workflow capability on Neo and SAP Build Process Automation.
- Runtime data, for example, workflow instances and task instances, aren’t migrated. Running workflow instances must be continued in the Neo tenant as well as completion of task instances.

3.4.7.1 Migrate Workflow Models

Store your workflow models in a multitarget application (MTA).

Context

The migration of workflow models consists of the following steps:

- Creating a new multitarget application project in SAP Business Application Studio.
- Creating a new *workflow module* inside the project.
- Moving your Neo workflow models and SAP UI5-based task UIs into the MTA project.

Procedure

1. Create a new multitarget application project with a workflow module in SAP Business Application Studio. For more information, see Create a Workflow Module [page 23].

   > ⓘ Note
   >
   > The subsequent steps refer to `workflow-module` as the name of your module. Replace this with the name of your workflow module accordingly.

2. Once the module has been created, delete the `workflow-module` folder using *Delete Permanently* from the context menu.
3. Rename the workflow project that you exported from SAP Web IDE Neo environment to `workflow-module.zip`.
4. Import the workflow project using *SAP Build Application Studio: Import Project* as a subfolder of your MTA project. For more information, about how imported projects are managed, see Importing Projects.
5. Copy or move the `workflow-module` folder of the imported project as a subfolder of your MTA project (this is the same folder level from which you previously deleted the folder).

   Your workflow models are available in the *workflows* folder, script task content in the *scripts* folder , and so on.

6. As a cross-check, open the `mta.yaml` file and make sure that it contains a module named `workflow-module`.
7. Switch to the *Resources* tab and verify that the service points to *process-automation-service*(that is, the parameter with key 'service' has the value 'process-automation-service').

Results

Your workflow models are now stored in a multitarget application. That is the deployment format for workflows in SAP Build Process Automation. If your workflows don’t contain SAP UI5-based task or start UIs, the project is ready for you to build and deploy. Otherwise, additional migration steps are required. First, the SAP UI5-based UIs must undergo additional migration steps. After this, the user tasks in your workflow models must be rebound to the migrated task UIs. You can put the task UIs either in the same MTA as the workflow module or in a separate one.

3.4.7.2 Migrate UI5-Based Task and Start UIs

Learn how to migrate SAPUI5 and start-based UIs to multitarget application.

Context

This section is only relevant for SAP UI5-based task and start UIs. If you’re using form-based UIs or no workflow UIs at all, you can skip this section.

Procedure

1. Create a new approuter configuration module as a prerequisite to enable SAP Build Work Zone, standard edition. If you don't want to put the task UIs into the same MTA with your workflow module, create another MTA project first. Otherwise, create the new module inside the MTA project you created. For more information, see Creating a Custom Task UI [page 112].
    a. Select *Managed Approuter*.
    b. Enter a meaningful business solution identifier. This is an additional identifier for your task UIs. Ideally, adhere to the namespace conventions of your company.
    c. Choose *Yes* confirming that you plan to add a UI. This adds a dependency to the HTML5 application repository to deploy your UI content to.
2. Once created, open the `mta.yaml` file and make sure that two additional modules are now available.
    - *workflow_mtar-app-content*
    - *workflow_mtar-destination-content*

3. Open the `manifest.json` file of the workflow UI you want to migrate from, that is, the Neo project. Use the `id` value of the `sap.app` property. In this procedure, let's assume that this value is `com.sap.workflow.samples.onbequip`.
4. Create a new module using the *Workflow UI* type inside the MTA project.
5. Enter a module name that matches the last segment of the app ID that you copied from your Neo project (here we use `onbequip`).
6. Depending on the type of workflow UI you want to migrate, select *Task UI* or *Start UI*.
7. Enter a namespace that matches all segments except for the last segment of the app ID. In our example, this then is `com.sap.workflow.samples`.
8. From the generated *Component.js* file, copy the functions *_fetchToken*, *getTaskInstanceID*, *_getTaskInstancesBaseURL*and *_getWorkflowRuntimeBaseURL* and save them to a text editor.
9. Select the following folders and use *Delete Permanently* to delete them:
    - *controller*
    - *css*
    - *i18n*
    - *model*
    - *view*
    - *Component.js*
10. From your Neo project, copy the content from the following folders into the *webapp* folder:
    - *controller*
    - *css*
    - *i18n*
    - *model*
    - *view*
    - *Component.js*
11. If you defined additional entities, such as data sources or models in the `manifest.json` in Neo, copy them over into the newly generated `manifest.json`.
12. URLs for HTTP requests to the back end must be adapted in the UI code. The most prominent example, the one for task completion, is for task completion which will be described here. Open the javascript file which initiates the backend call (For example,. in many cases it is located in `Component.js` or `Controller.js`.
13. Paste the functions you have copied earlier into the file.
14. Search for occurrences where URLs for backend calls to the SAP Build Process Automation API are triggered (e.g. indicated by `$.ajax`):
    a. Replace the code constructing the URLs with the functions *_getTaskInstancesBaseURL* and *_getWorkflowRuntimeBaseURL* respectively.
    b. Replace retrieval of *xsrf* tokens with the function *_fetchToken*.

    The following is an example of the code used:

    Example

    Sample Code

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

15. If you have additional back-end calls, change the relative path in the `Component.js` or `Controller.js`as outlined in step 12, which mentions about how URLs for HTTP requests must be adapted to UI code.
16. You might have defined additional configurations, for example, routes in the `neo-app.json` file to call other services than the workflow back end. Redefine these configurations in the `xs-app.json` file of the task UI module. See Neo - Cloud Foundry Application Distractor Mapping in the official migration guide of SAP UI5.

3.4.7.3 Rebinding Your Task UIs in Your Workflow Models

After you copied the workflow models and task UIs into an MTA project, you need to rebind all user task SAPUI5-based activities to the corresponding UIs. Form-based user tasks don't need to be migrated.

Procedure

1. In SAP Business Application Studio, open the workflow model in the workflow editor.
2. Select the user task you want to rebind.
3. From *User Task Properties*, choose *User Interface*.
4. Choose *Select*.

   The dialog that opens displays all UI modules inside the MTA project that you've created.
5. Select the UI module that matches the current user task. The value for *Application Name* is selected automatically.
6. In *SAPUI5 Component Path*, select *webApp*.
7. Choose *OK* to confirm your selections.

   This updates the *User Interface* section and rebinds the UI.
8. As a cross-check, verify that the *Business Solution Name* field matches the value that you initially entered when creating the approuter configuration. In addition, check that the error marker on the user task in the workflow model has gone away.
9. Repeat the above steps for all user tasks in all migrated workflow models.

Results

You've migrated the workflow models and workflow UIs to an MTA-based project structure that is suitable for deployment to SAP Build Process Automation.

3.4.7.4 Build and Deploy the MTAR

Build and deploy your multitarget application to the SAP BTP, Cloud Foundry environment.

Procedure

1. To build the MTAR file, open the context menu *Build MTA Project* of the `mta.yaml` file.

   After a successful build, the built archive is available in the *mtar_archives* folder.
2. To deploy the MTAR file, from the context menu of the built archive, choose *Deploy*. If not yet done, configure the Cloud Foundry coordinates and credentials of the subaccount in which SAP Build Process Automation is subscribed.
3. After deploying the MTAR, go to the SAP BTP cockpit, and navigate to the subaccount to which you deployed the MTAR file.
4. From the navigation pane, choose *HTML5 Applications*.
5. Make sure that the UI5 apps that you bundled in the MTAR file are listed with the *Business Solution Name* you have chosen.
6. Log in to SAP Build Process Automation, and choose *Monitor* > *Manage* > *Process and Workflows*.
7. In the search field, enter the name of one of your workflow models and make sure that they can be found.
8. If your MTAR file contains a workflow start UI, see Create Custom Start UI Tiles on the Central Launchpad [page 239].

4 Consuming the Workflow Capability at Application Runtime

This section describes the tasks for the workflow capability administrator and contains a user guide for business users of the workflow capability.

Related Information

Configuring the Workflow Capability [page 190]
Using the Workflow Capability [page 243]

4.1 Configuring the Workflow Capability

Before you can use the workflow capability, meet the prerequisites and execute the basic setup.

Prerequisites

Initial Setup [page 17]

Procedure

1. Create a service instance of the workflow capability.

   You can either create it in the cockpit or using the command-line interface:

   - Create a Service Instance of the Workflow Capability Using the Cockpit [page 191]
   - Create a Service Instance of the Workflow Capability Using the Command Line Interface [page 193]
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 258], 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 91].
   - 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 209].

4.1.1 Create a Service Instance of the Workflow Capability Using the Cockpit

You can create an instance of the workflow capability 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 capability 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 261].

> ⚠ Caution
>
> As soon as you delete the last remaining service instance of the workflow capability across all the spaces in your organization, all your data within the workflow capability is irrevocably erased.

However, if at the time of deletion an SAP Build Process Automation instance exists, the data will not be deleted and stays available for use in SAP Build Process Automation.

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 is set to *standard*, then enter a name for the instance, and choose *Next*.
6. (Optional) Provide a JSON object that may contain one or multiple of the following configurations.
    - a list of authorities to get granted to the OAuth client. For more information, see Technical Authentication [page 261].
    - a list of redirect URIs that SAP BTP checks while redirecting. If your landscape domain or custom domain aren't on this list, including wildcards, the SAP Authorization and Trust Management Service won't redirect the users. For more information, see General SAP BTP Application Security Documentation.

    Example

    ```
    {
        "authorities": [
            "WORKFLOW_INSTANCE_START",
            "MESSAGE_SEND"
        ],
        "xs-security": {
            "oauth2-configuration": {
                "redirect-uris": ["https://
    myapp.cfapps.eu10-004.hana.ondemand.com", "https://myapp.mydomain.com/my/
    content"]
            }
        }
    }
    ```
7. Choose *Next*.
8. Check the overview, and then choose *Create*.

    The new instance appears on the *Instances* 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 the *Instances* 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 inline.

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 the Workflow Capability Using the Command Line Interface

You can create an instance of the workflow capability 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 the Workflow Capability 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 capability across all the spaces in your organization, all your data within the workflow capability is irrevocably erased.

Procedure

1. Check that the workflow capability is available on the marketplace:

   ```
   cf marketplace
   ```

   You should see a service named the workflow capability with a plan named *standard*.
2. Create a service instance of the workflow capability:

If you're using a global account, execute the following command:

```
cf create-service workflow standard <instance-name>[-c <parameters-as-json>]
```

Where, the `<instance-name>` is the service instance name of your choice, and the optional `-c` parameter, the `<parameters-as-json>` is a JSON object that may contain one or multiple of the following configurations:

- a list of authorities to get granted to the OAuth client. For more information, see Technical Authentication [page 261].
- a list of redirect URIs that SAP BTP checks while redirecting. If your landscape domain or custom domain aren't on this list, including wildcards, the SAP Authorization and Trust Management Service won't redirect the users. For more information, see General SAP BTP Application Security Documentation.

> Example
>
> ```
> {
>     "authorities": [
>         "WORKFLOW_INSTANCE_START",
>         "MESSAGE_SEND"
>     ],
>     "xs-security": {
>         "oauth2-configuration": {
>             "redirect-uris": ["https://
> myapp.cfapps.eu10-004.hana.ondemand.com", "https://myapp.mydomain.com/my/
> content"]
>         }
>     }
> }
> ```

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 the previous step.

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 the step where you created a service instance 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 detect the parameters of a service instance.

Procedure

Check the current configuration of a service instance of workflow capability 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 the workflow capability instances that are directly created in a subaccount, as well as those created indirectly through subscription to an application that uses the workflow capability, 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 `defaultCollectionQueryFilter` service instance parameter and the respective `shared` or `own` (default) value.

> ⓘ Note
>
> - Workflow data that originates from a subscription to an application is deleted once the application is unsubscribed.
> - Applications that use the workflow capability 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 workflow capability 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 the Workflow Capability Using the Cockpit [page 191] and Create a Service Instance of the Workflow Capability Using the Command Line Interface [page 193].
   - 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 that were initially created without a JSON object.
     See Update Service Instances Using the SAP BTP Cockpit or Cloud Foundry Command Line Interface.

Related Information

Retrieve the Configuration of a Service Instance [page 195]

4.1.4 Enable Technical Authentication

Access the workflow capability 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 workflow capability SAP Business Accelerator 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 the WORKFLOW_INSTANCE_START and MESSAGE_SEND authorities to the service instance:

   Sample Code

   ```
   {
       "authorities": [
           "WORKFLOW_INSTANCE_START",
           "MESSAGE_SEND"
       ]
   }
   ```

3. Create or update a service instance for the workflow capability with necessary scopes and provide the JSON object.

   List of possible scopes:

   - WORKFLOW_INSTANCE_START
   - WORKFLOW_DEFINITION_GET
   - WORKFLOW_INSTANCE_GET
   - WORKFLOW_INSTANCES_UPDATE
   - WORKFLOW_INSTANCE_CANCEL
   - WORKFLOW_INSTANCE_GET_ERROR_MESSAGES
   - WORKFLOW_INSTANCE_GET_CONTEXT
   - WORKFLOW_INSTANCE_GET_EXECUTION_LOGS
   - MESSAGE_SEND
   - TASK_GET

   - Create a new service instance
     Provide the JSON object as parameter while creating a new service instance. See Create a Service Instance of the Workflow Capability Using the Cockpit [page 191] and Create a Service Instance of the Workflow Capability Using the Command Line Interface [page 193].
   - 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 that 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 161].

Related Information

Technical Authentication [page 261]

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.

> ⓘ Note
>
> Technical authentication 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. For more information, see Technical Authentication [page 261].

Before you can use principal propagation, the following one-time configuration is required:

Create a destination that uses the OAuth credentials of the configuration parameters for the workflow capability. For more information, see Create an OAuth Destination [page 198].

You need one destination for each service that is called. For more information, see:

- Configure Service Tasks [page 46]
- Destinations [page 262]

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

Log in to the cockpit and open the *Destinations* editor. See Access the Destinations Editor.

Context

To configure destinations, use the standard SAP Business Technology Platform (SAP BTP) mechanisms in the SAP BTP cockpit. For more information, see Managing Destinations.

Procedure

Create an OAuth destination for the workflow capability runtime using the following values:

Field	Value
Name	**`bpmworkflowruntimeoauth`**
Type	HTTP
Description	Enter a text, for example, **`Workflow capability 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 capability. This corresponds to the URL in the `uaa.url` parameter as described in Determine Service Configuration Parameters [page 158].
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 158].
Password	Set the password to the secret. See the `uaa.clientsecret` parameter as described in Determine Service Configuration Parameters [page 158]. ⚠ 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 Capability Mail Destination

Before you can send notification mails for mail tasks within a workflow, you must first configure a mail destination.

Prerequisites

- You have configured a mail destination. For more information, see Configure an SMTP Mail Destination.
- You have the details for configuring an SMTP email for your scenario.

- To use OAuth authentication, you have collected the details for configuring the authentication retrieval, that is, client ID and client secret as well as the scopes. For more information about scopes, see the Guided Answers.
- Your mail server has the following characteristics:
  - It supports the `SMTP STARTTLS` command on port 587, because the workflow capability supports only STARTTLS on this port. This means that you must only use a configuration according to SMTP Message Submission (RFC 4409). Message relay configurations, typically using port 25, are not permitted.
  - It requires authentication, because the workflow capability doesn't support unauthenticated logins.
  - It has enabled support for Basic Authentication or for the SASL XOAUTH2 authentication mechanism, because other authentication types aren't supported.
- To use the XOAUTH2 mechanism, your OAuth2 authentication server must have the following characteristics:
  - The authorization server must support the OAuth2 Password Grant flow, also known as Resource Owner Password Credentials Grant.
  - The authorization server must have the Password Grant flow and other prerequisite configurations for authenticated SMTP enabled for the mailbox and account that are represented by the login credentials.
  - The authentication server must not require user interaction during token retrieval, for example, for multifactor authentication or for delegation to federated authentication servers.
  - Prerequisite configuration for authenticated SMTP have been enabled for the mailbox, account and OAuth2Authentication. For OAuth2ClientCredentials flow within Microsoft 365 see Register a Service Principal for a Microsoft Azure Application [page 205] .
  - Password grant flow and other prerequisite configurations for authenticated SMTP have been enabled for the mailbox and account represented by the login credentials.

  > ⓘ Note
  >
  > If you use Microsoft Azure, you must an configure Azure application for SMTP via Microsoft 365. For more information, see Configure Azure Application for SMTP Authentication [page 204].

  > ⓘ Note
  >
  > We recommend that you use OAuth2 authentication over Basic authentication if your mail server supports both, due to security considerations and industry best practices. When you apply OAuth2 authentication we recommend that you use OAuth2 Client Credentials Flow over OAuth2 Password Grant Flow, if your mail server supports both.
- 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 Members to Your Global Account.
  Also, security administrators (who 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.
- In the SAP BTP, Cloud Foundry environment, you must have authorizations for maintaining destinations in the respective subaccount, as described in What Is SAP BTP Connectivity? in the SAP Connectivity service documentation.

> ⓘ Note
>
> For more information about authentication and connection failures on mail tasks, see the Guided Answers.

Context

For troubleshooting any issues, see Authentication and Connection Failures on Mail Tasks in the Guided Answers.

Procedure

Create a destination in the SAP BTP cockpit.

For more information, see Managing Destinations.

> ⓘ Note
>
> The *OnPremise* proxy type is not supported. That is, mail servers that can only be reached using a Cloud Connector, are not supported.

- To import a destination:
  1. Save the template as a file.

     ```
     Type=MAIL

     Name=bpmworkflowruntime_mail
     Authentication=BasicAuthentication

     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.
  3. Adapt the authentication according to your requirements. Only `BasicAuthentication`, `OAuth2Password` and `OAuth2ClientCredentials` are supported for the *Authentication* field. For `OAuth2Password` and `OAuth2ClientCredentials` additional properties, as indicated by the editor, are mandatory. If you use an OAuth2-related authentication, specify the following property to enable

SASL XOAUTH2 support: Set `mail.smtp.auth.mechanisms` to **`XOAUTH2`**. In many cases, you must also configure the `scope` property as a list of space-delimited, case-sensitive strings. For information about the configuration, see HTTP Destinations in the SAP Connectivity service documentation.

- 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`**
Proxy Type	Internet
User	User for logging in to the mail server
Password	Password for logging in to the mail server
Authentication	The authentication method required. The methods `BasicAuthentication`, `OAuth2ClientCredentials` or `OAuth2Password` are supported. For `BasicAuthentication`, `NoAuthentication` is possible if `mail.bpm.send.disabled` is set to true. This setting turns off the interaction with the mail server. For example, it's used temporarily while you define a workflow. For `OAuth2Password`, additional properties for the OAuth authentication, as indicated by the editor, are mandatory. In many cases, you must also configure the `scope` property as a list of space-delimited, case-sensitive strings. See the SAP Connectivity service documentation.
If you're using `OAuth2Password` or `BasicAuthentication`, the following additional values are required:	
User	The user for logging into the mail server
Password	The password for logging into the mail server
The following additional properties are required if you're using an `OAuth2` authentication method:	
mail.smtp.auth.mechanisms	XOAUTH2
scope	Value for the OAuth 2.0 scope parameter, expressed as a list of space-delimited, case-sensitive strings

Property	Value
`mail.transport.protocol`	**`smtp`**
`mail.smtp.auth`	**`true`**
`mail.smtp.starttls.required`	**`true`**

Property	Value
`mail.smtp.host`	Host name of your mail server
`mail.smtp.port`	Port on which your mail server listens for connections (typically **`587`**), other ports, such as 465 are not supported.
`mail.smtp.from`	Mail address to use as the "From" address of mails sent by the workflow capability, for example, **`cpworkflow@example.com`**. This address must belong to an existing mailbox because it receives the replies to mails that the workflow capability 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. Therefore, only consider setting this property if you use self-signed certificates or certificates signed by less-common root CAs. For more information, see Determine the Trusted CAs of SAP JVM Truststore [page 205].
`mail.bpm.send.disabled`	• **`true`** Turns off interaction with the mail server, for example, temporarily while you develop a workflow. • **`false`**
`mail.smtp.auth.mechanisms`	**`XOAUTH2`** If you use an OAuth2-related authentication, set this property to enable SASL XOAUTH2. Otherwise, leave it empty.

For more information about the properties, see the JavaMail API documentation.

> ⓘ Note
>
> Only these mail-related properties are evaluated. Other mail-related properties that are configured in the mail destination have no effect. However, for the optimal operation of the mail functionality, the workflow capability might apply additional properties, such as connection timeouts. For the OAuth2-related authentication types, the properties documented by the SAP Connectivity service are also supported.

Related Information

Guided Answers

4.1.6.1 Configure Azure Application for SMTP Authentication

You can configure an Azure application for SMTP authentication, by doing this it acts as an OAuth2 client, which is required to create client credentials and use the OAuth2 protocol for authentication.

Prerequisites

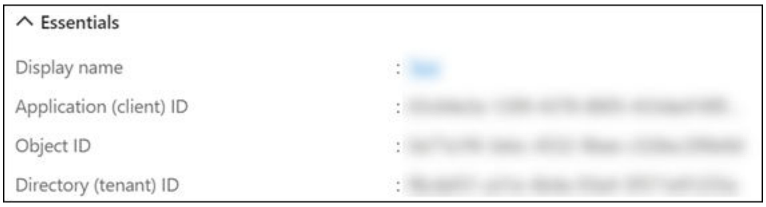
You have access to the Azure application.

Procedure

1. Create a client credential. A client secret is defined by a description, an expiration date, a value, and a secret ID. The client secret value is masked later, and you can't retrieve the value if you don't save it at this step. Therefore, click the copy icon beside the value of the new client secret and paste it somewhere from where you can retrieve it later as the client secret. To create a client credential for your Azure application, see Add a Client Secret .

> ⓘ Note
>
> You must set *Accounts in any organizational directory* for supported account types.

2. Find relevant IDs that define the application. From the *Overview* page of your Azure application, you can find the *Client ID* in *Application (client) ID* field and the *Tenant ID* in *Directory (tenant) ID* field.



^ Essentials

Display name : [redacted]

Application (client) ID : [redacted]

Object ID : [redacted]

Directory (tenant) ID : [redacted]

3. Complete the registration for OAuth2 Client Credentials flow without defining a redirect URL.

4.1.6.2 Register a Service Principal for a Microsoft Azure Application

You can register a service principal for a Microsoft Azure application. This is necessary to grant access to mailboxes when using OAuth2 client credential authentication.

Procedure

1. Add SMTP permissions to your Microsoft Entra application. For more information, see Authenticate an IMAP, POP or SMTP connection using OAuth ➦ .
2. Register service principals in Microsoft Exchange once a tenant ID is approved by Microsoft Entra application. For more information, see Register service principals in Microsoft Exchange ➦ .
3. Confirm SMTP auth is enabled for the mailbox. For more information, see Enable SMTP AUTH for specific mailboxes ➦ .

Results

You have registered a service principal for a Microsoft Azure application.

4.1.6.3 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 `mail.smtp.ssl.trust` trust list property.

The workflow capability 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 relative to the main folder of the SAP JVM at `$JAVA_HOME/jre/lib/security/cacerts`.
    b. Determine the location of the keytool program of your installation. Typically, it's located relative to the main folder of the SAP JVM at `$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 is usually **`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 Capability Attachments

The workflow capability provides a lightweight integration with the SAP Document Management service. This means that binary files are completely handled by the SAP Document Management service (storage, access management, and more) while the workflow capability maintains the relation between a workflow instance, its files, and additional metadata.

Prerequisites

- You have set up the SAP Document Management service (integration option).
- You have connected a compliant repository supporting CMIS 1.1 browser binding.
  For more information about the SAP Document Management service and the setup, see SAP Document Management service.

Configure an Attachment 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

Attribute	Value
URL	`<repository-base-url>/browser/<repository-id>` Note Make sure that the URL points to a repository without `/root/`. Otherwise, you'll run into a 405 error.
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 the 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 165]

4.1.8 Configure an SAP Business Accelerator Hub Destination

Before you search for the workflow capability APIs in SAP Business Application Studio, you must first configure a destination for SAP Business Accelerator Hub.

Prerequisites

In the SAP BTP, Cloud Foundry environment on subaccount level, you must be a global account member to manage destinations. See Add Members to Your Global Account.

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 Business Accelerator Hub
Type=HTTP
TrustAll=true
HTML5.DynamicDestination=true
Authentication=NoAuthentication
WebIDEUsage=apihub_catalog
Name=SAP_Business_Accelerator_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_Business_Accelerator_Hub**
Type	*HTTP*
Description	Text that describes the destination, for example, **Destinaton for SAP Business Accelerator Hub**
URL	**https://api.sap.com:443**
Proxy Type	Internet
Authentication	NoAuthentication

Property	Value
HTML5.DynamicDestination	**true**
TrustAll	**true**
WebIDEEnabled	**true**
WebIDEUsage	**apihub_catalog**

> ⓘ Note
>
> Only the above properties are evaluated. Other properties that are configured in the destination have no effect. However, for the optimal operation of the functionality, the workflow capability 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 Build Work Zone, standard edition and its functionalities.

Prerequisites

- You have subscribed to the Launchpad application and configured the necessary roles for your user in your subaccount. See Initial Setup in the SAP Build Work Zone, standard edition documentation.
- You have a workflow capability instance running. See Create a Service Instance of the Workflow Capability Using the Cockpit [page 191].

- You have created a destination to the workflow capability instance with the OAuth2JWTBearer authentication method. See Destinations Pointing to Service Instances.
- You have a site. See the Create a Site.

Context

> ⓘ Note
>
> Only workflows with custom task UIs that use the Managed Approuter can be used in the central site. 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 219].

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 site, choose *Provider Manager*.
4. To make the default *HTML5 Apps* content provider load the standard apps of the workflow capability, 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 items:
    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 change 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. Do not maintain a destination name in the *System* field.

    d. Configure the locally created copies as described in the following sections:
    - My Inbox: Configure My Inbox App [page 211]

    - Monitor Workflows: Configure the Monitor Workflows App [page 215]
    - BPM Form Player: Configure a Start-Form-Based Workflow Start App [page 215]
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 Build Work Zone, standard edition documentation.
10. Access the site. 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 209].

Context

Each display option needs its 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 Build Work Zone, standard edition, 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 enable or disable the *Expert View* or other functionalities, you have to manually set parameters for them.
    - *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 246].	`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, use 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`

- 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 absence by creating substitution rules for planned and unplanned absences. For more information, see Create and Manage Substitution Rules [page 250]. Always use this parameter together with `userSearch`.	`true`	`false` or `true`

Parameter Name	Usage	Default Value	Permitted Values
`userSearch`	Allows you to enter a user ID when creating a substitution rule. The `userSearch` value must be set to `false`.	`true`	`false`

- Forward

Parameter Name	Usage	Default Value	Permitted Values
`userSearch`	Allows you to enter a user ID when forwarding a task. The `userSearch` value must be set to `false`. For more information, see Configure User Tasks [page 29].	`true`	`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

- *Show Log* button

Parameter Name	Usage	Default Value	Permitted Values
`showLog`	Controls the visibility of the *Show Log* button.	`true`	`false` or `true`

- *Confidence Level* column in the *Expert View*

Parameter Name	Usage	Default Value	Permitted Values
`confidenceLevel`	Controls the visibility of the *Confidence Level* column in the task table of the *Expert View*.	`false`	`false` or `true`

- Pagination

Parameter Name	Usage	Default Value	Permitted Values
`enablePaging`	Enables client-size paging in My Inbox in *Master-Detail View*.	`false`	`true` or `false`
`pageSize`	The number of items to be loaded in one	30	Integer

Parameter Name	Usage	Default Value	Permitted Values
	page. Must be set with `enablePaging`.		
`pageSizeServiceLimit`	(Optional) This parameter should only be used if you have a high number of tasks in My Inbox. It specifies the number of records you want to show in My Inbox. You can see at most 1000 records per API call. This parameter can't be used in the *Expert View*. ⚠ Caution Please note, that this parameter leads to change in the user interaction in My Inbox. After you choose a finalizing action (for example, *Approve* or *Reject*) or a non-finalizing action (for example, *Claim* or *Forward*), the selection will move to the first available task in the list.	`1000`	The value must correspond to the maximum allowed records for the Workflow Capability API call. For more information, see *GET:/TaskCollection* in Inbox API for Cloud Foundry.

> ⚠ Caution
>
> For the proper usage of My Inbox, do not change the parameters on the *Navigation* tab, which are not listed above.

4. Save your changes.

4.1.9.2 Configure the Monitor Workflows App

The Monitor Workflows app can display both workflow definitions and workflow instances.

Prerequisites

You have created at least one local copy of the Monitor Workflows app. See Create Workflow and My Inbox Tiles on Central Launchpad [page 209].

Context

Each display option needs its own local copy of the Monitor Workflows app. The local copies of the apps must be configured accordingly.

Procedure

1. In the *Content Manager* of the central SAP Fiori launchpad, open the local copy of the Monitor Workflows app.
2. Go to the *Navigation* tab of the local app, 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 137].
- 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 209].

Context

You always need to 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 its 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 choose *Edit*.
3. Set the parameters by adding new parameters that match the parameters 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, business users can work only on tasks, which are of specific, preconfigured task types. But 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 209].

Context

With the workflow capability, 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 254].

Procedure

1. In the navigation area of the central SAP Build Work Zone, standard edition, 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 the parameter and a comma-separated list of `TaskDefinitionIDs` as the value.

     > → Tip
     >
     > To get the `TaskDefinitionID`, follow these steps:
     >
     > 1. From the *All Items Tile* in My Inbox, select the task that you want 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 this:

       ```
       /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:

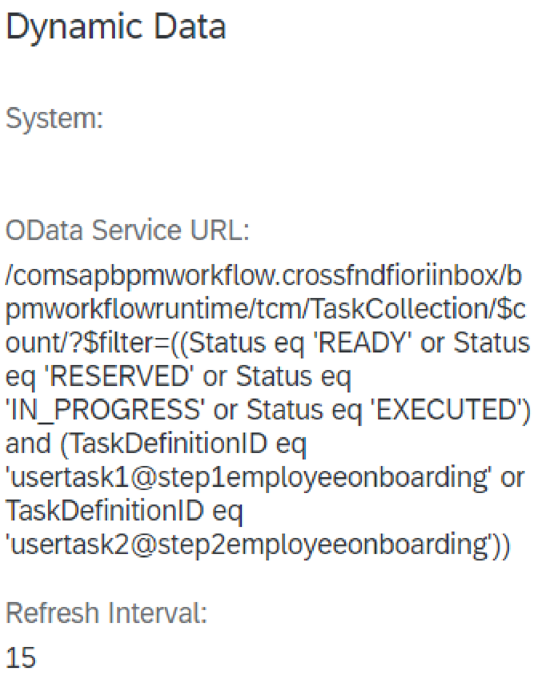
       ```
       /bpmworkflowruntime/odata/v1/tcm/TaskCollection/$count/?
       $filter=((Status eq 'READY' or Status
       ```

```
eq 'RESERVED' or Status eq 'IN_PROGRESS' or Status eq 'EXECUTED')
and (TaskDefinitionID eq 'usertask1@step1employeeonboarding'
or TaskDefinitionID eq 'usertask2@step2employeeonboarding'))
```

> ⓘ Note
>
> The list of **`TaskDefinitionIDs`** in the newly added *OData Service URL* task-based filter must match the **`TaskDefinitionIDs`** in the **`TaskDefinitions`** parameter, so that the tile counter of the scenario-specific tile matches 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 like this:



Dynamic Data

System:

OData Service URL:

/comsapbpmworkflow.crossfndfioriinbox/bpmworkflowruntime/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'))

Refresh Interval:

15

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 SAP Fiori launchpad functionalities. Follow the instructions to create a dedicated SAP Fiori launchpad module, to build and deploy it.

Context

> ⓘ 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 209].

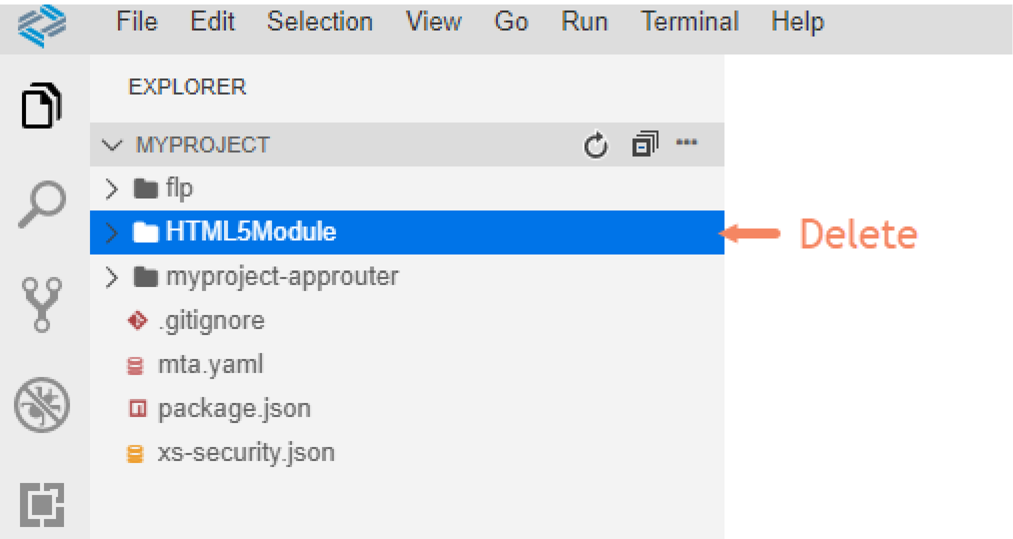
Using SAP Business Application Studio

Procedure

1. Create a multitarget 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.

   > ⓘ Note
   >
   > `<MTA_ID>` is 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.

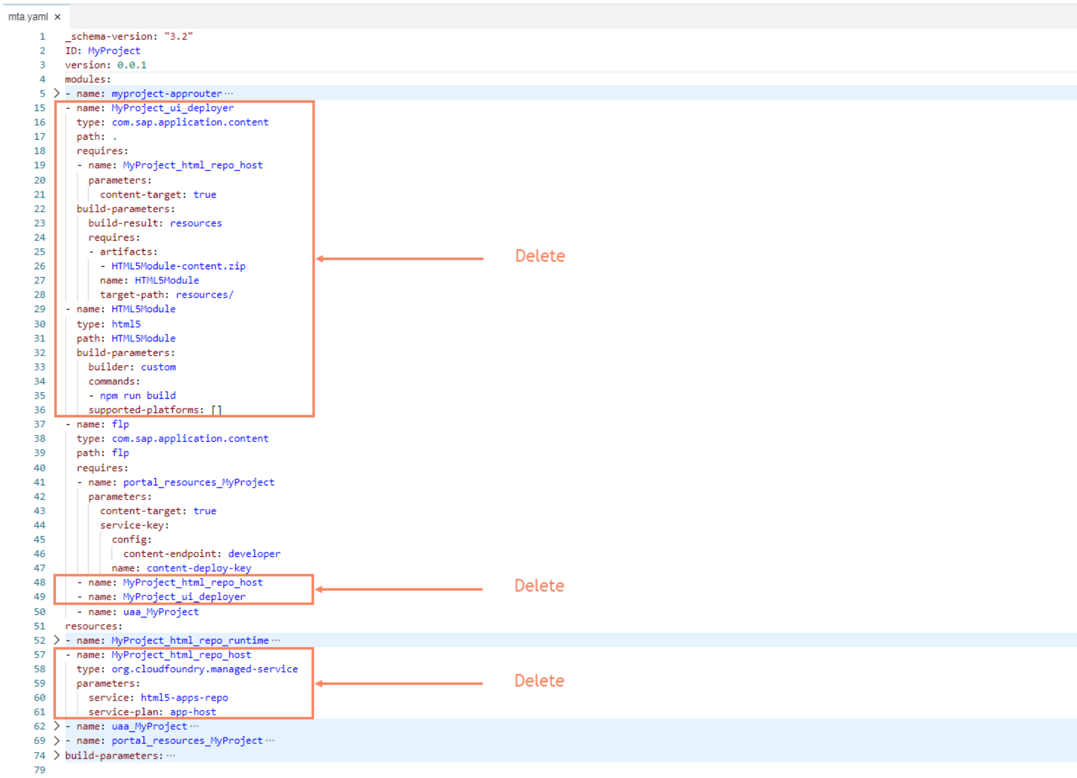
2. Delete the `HTML5Module` folder located in your project. The folder might be named differently depending on the name that you used in the previous step. Execute the action by right-clicking the folder and choosing *Delete*.



3. Right-click the `mta.yaml` file located in your project and select *Open With* > *Code Editor*.

> ⚠ Caution
>
> While you are working 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 that you 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.



```
_schema-version: "3.2"
ID: MyProject
version: 0.0.1
modules:
- name: myproject-approuter···
- name: MyProject_ui_deployer
  type: com.sap.application.content
  path: .
  requires:
  - name: MyProject_html_repo_host
    parameters:
      content-target: true
  build-parameters:
    build-result: resources
    requires:
    - artifacts:
      - HTML5Module-content.zip
      name: HTML5Module
      target-path: resources/
- name: HTML5Module
  type: html5
  path: HTML5Module
  build-parameters:
    builder: custom
    commands:
    - npm run build
    supported-platforms: []
- name: flp
  type: com.sap.application.content
  path: flp
  requires:
  - name: portal_resources_MyProject
    parameters:
      content-target: true
      service-key:
        config:
          content-endpoint: developer
        name: content-deploy-key
  - name: MyProject_html_repo_host
  - name: MyProject_ui_deployer
  - name: uaa_MyProject
resources:
- name: MyProject_html_repo_runtime···
- name: MyProject_html_repo_host
  type: org.cloudfoundry.managed-service
  parameters:
    service: html5-apps-repo
    service-plan: app-host
- name: uaa_MyProject···
- name: portal_resources_MyProject···
build-parameters:···
```

2. Add a dependency to the workflow capability in the `requires` section of your project.

> ⓘ Note
>
> - Verify that you have provided correct indentations using spaces in the `mta.yaml` file.
> - `<workflow_capability_instance_name>` is the workflow capability instance name created in the cockpit.
> - `<SAP Fiori launchpad 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_capability_instance_name>
    - name: uaa_<MTA ID>
```

3. Add a dependency to the workflow capability instance in the `resources` section of your project:

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

4. Add a `uaa` dependency:
    1. If there's no `xs-security.json` file available, then create an `xs-security.json` file at the same level as the `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 a dependency to the workflow capability 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_capability_instance_name>
    - name: uaa_<MTA ID>
```

4. In the `xs-app.json` file, inside the `<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 Build and Deploy.
7. Add the required roles to users. For more information, see Authorization Configuration [page 258].
8. Open SAP Fiori launchpad.
   You can now see the My Inbox, *Monitor Workflows (Workflow Instances)*, and *Monitor Workflows (Workflow Definition)* tiles.
9. (Optional) Customize your My Inbox app tile. For more information, see Configure My Inbox App [page 211].

Using SAP Web IDE

Prerequisites

1. Create a service instance of workflow capability. For more information, see:
   - Create a Service Instance of the Workflow Capability Using the Cockpit [page 191]
   - Create a Service Instance of the Workflow Capability Using the Command Line Interface [page 193].
2. 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 91].
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. Create a Multitarget Application and Add Modules
4. (Optional) To use an existing instance of the workflow capability, define it by using the `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 capability by navigating to ▶ *Services* ▶ *Service Instances* ▶ in the cockpit. Replace all occurrences of the existing instance name.

Sample Code

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

Make sure to replace the instance name in the modules sections as well.

5. Add an SAP Fiori Launchpad Module
6. Add the tiles on the SAP Fiori launchpad, for example, to your default group.

Sample Code

```
{
  "id": "cross.fnd.fiori.inbox",
  "appId": "cross.fnd.fiori.inbox",
  "vizId": "WorkflowTask-DisplayMyInbox"
}, {
  "id": "com.sap.bpm.monitorworkflow",
  "appId": "com.sap.bpm.monitorworkflow",
  "vizId": "bpmworkflowmonitor-DisplayInstances"
}, {
  "id": "com.sap.bpm.monitorworkflow",
  "appId": "com.sap.bpm.monitorworkflow",
  "vizId": "bpmworkflowmonitor-DisplayDefinitions"
}
```

7. Build and deploy your project. For more information, see Packaging and Deploying Applications to Production Systems.
8. Add the required roles to users. For more information, see Authorization Configuration [page 258].
9. 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.
10. (Optional) Customize your My Inbox app tile. For more information, see Configure My Inbox App [page 211].

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 customers after January 15, 2021, see Configure My Inbox App [page 211].

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 Adding a Custom Visualization (with SAP Web IDE). The table below provides an overview of the personalization parameters supported by My Inbox:

Parameter Name	Use	Default Value	Permitted Values
`expertMode`	Enables the *Expert View* of My Inbox.	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 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 `expertMode` parameter enables you to use the *Expert View* of My Inbox. For more information, see Expert View [page 246].

The `showAdditionalAttributes` parameter 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 `substitution` and `userSearch` parameters 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 250].

> ⓘ 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 231].

The `userSearch` parameter also 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 My Inbox features:

- Expert View [page 246] of My Inbox
- Display Custom Attributes in My Inbox [page 41]

Using this configurational approach, you can do the following:

- Create and Customize the Existing My Inbox Tile with SAP Business Application Studio [page 226]
- Create and Customize a New My Inbox Tile with SAP Web IDE [page 231]
- Create and Manage Substitution Rules [page 250]

4.1.9.5.1.1 Create and Customize the Existing My Inbox Tile with SAP Business Application Studio

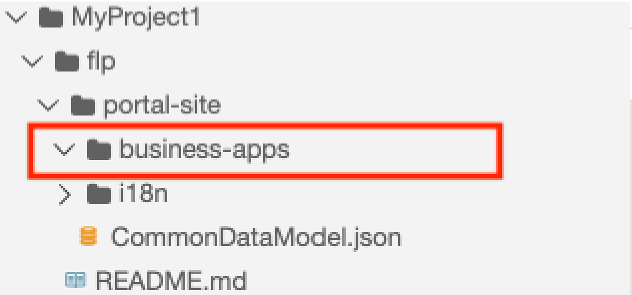
This procedure changes the *Master-Detail* layout of your existing My Inbox tile into the *Expert View* layout, which displays custom attributes that are configured during design time.

Context

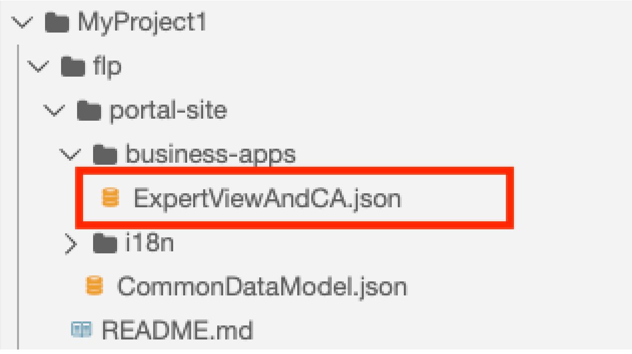
To create a new My Inbox tile, perform steps 1–8 as described in Create Workflow and My Inbox Tiles on Central Launchpad [page 209] and continue with the procedure below to customize the tile.

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 that 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 `expertMode` and `showAdditionalAttributes` parameters to **`"true"`** in order to activate expert mode or the custom attributes, respectively. These parameters are not interdependent. Depending on your requirements, 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 `expertMode`, make sure you remove the following from the JSON file:

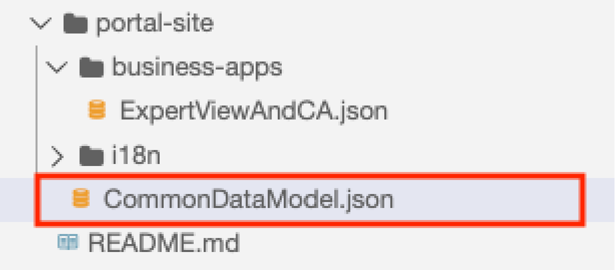
  ```
  "expertMode":{
      "value":{
          "value":"true",
          "format":"plain"
      }
  },
  ```
- If you don’t want to activate the custom attributes, make sure you remove the following from the JSON file:

  ```
  "showAdditionalAttributes":{
      "value":{
          "value":"true",
          "format":"plain"
      }
  },
  ```

4. Add a new My Inbox tile on the SAP Fiori launchpad.

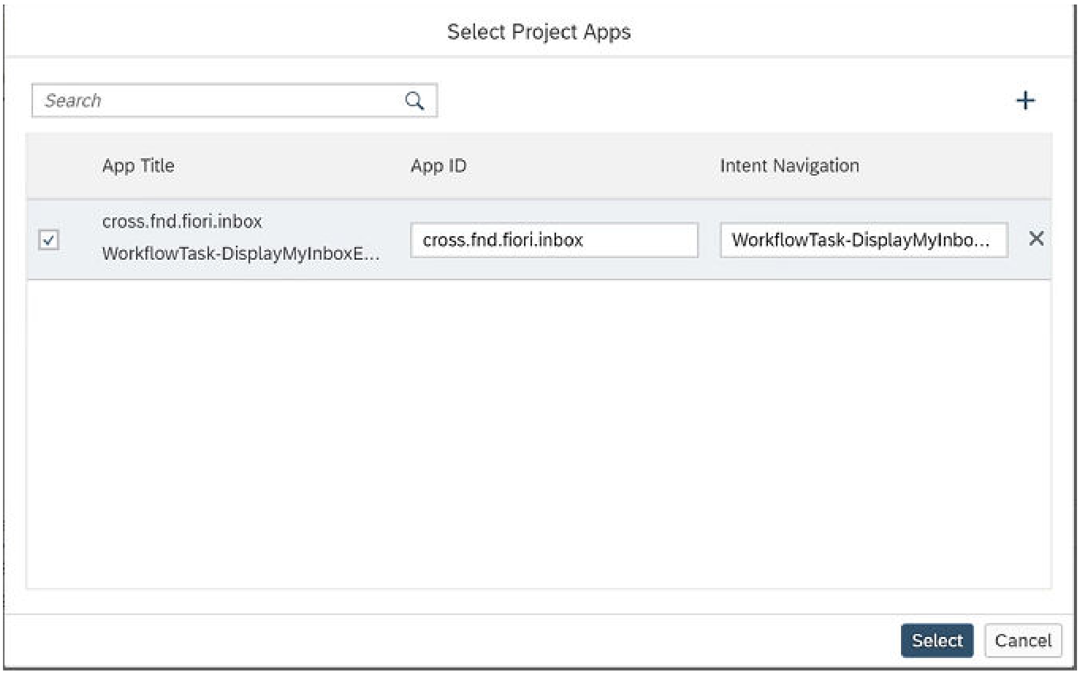
   You can create and customize a new My Inbox tile on the SAP Fiori launchpad by using the *Launchpad Editor* or the *Code Editor*:

   1. Using the *Launchpad Editor*:
      - In the *Launchpad Editor* under the ▶ *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`



Select Project Apps

	App Title	App ID	Intent Navigation
☑	cross.fnd.fiori.inbox WorkflowTask-DisplayMyInboxE...	cross.fnd.fiori.inbox	WorkflowTask-DisplayMyInbo...

- Choose *Select* to finish.

2. Using the *Code editor*:

- In the *Launchpad Editor* under the *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, perform 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 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"
            }
         ]
      }
   }
]
```

> ⓘ 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 and Deploy and Building and Deploying Multitarget Applications.

4.1.9.5.1.2 Create and Customize a New My Inbox Tile with SAP Web IDE

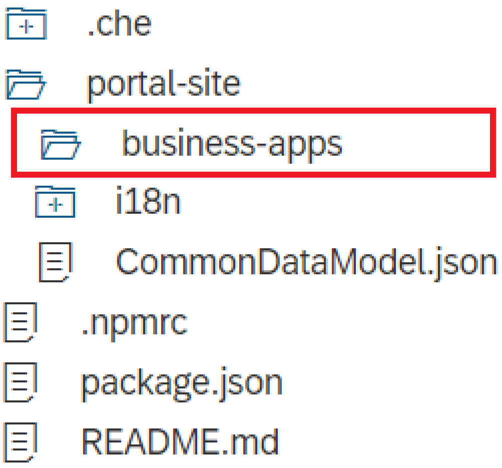
Follow this procedure to create a new My Inbox tile with the *Expert View* layout that will display custom attributes.

Context

To create a new My Inbox tile, perform steps 1–8 as described in Create Workflow and My Inbox Tiles on Central Launchpad [page 209] and continue with the procedure below to customize the tile.

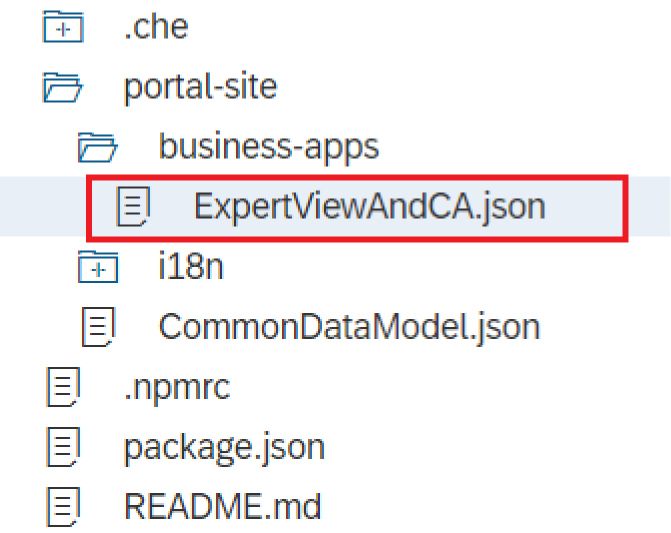
Procedure

1. Create a folder called `business-apps` in the `portal-site` folder of the SAP Fiori launchpad module.



```
.che
portal-site
  business-apps
  i18n
  CommonDataModel.json
.npmrc
package.json
README.md
```

2. Create a JSON file within the `business-apps` folder and give it a name.



.che
portal-site
business-apps
ExpertViewAndCA.json
i18n
CommonDataModel.json
.npmrc
package.json
README.md

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 `expertMode` and `showAdditionalAttributes` parameters to **`"true"`** in order to activate expert mode or the custom attributes respectively. These parameters are not interdependent. Depending on your requirements, you can only set 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 `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:

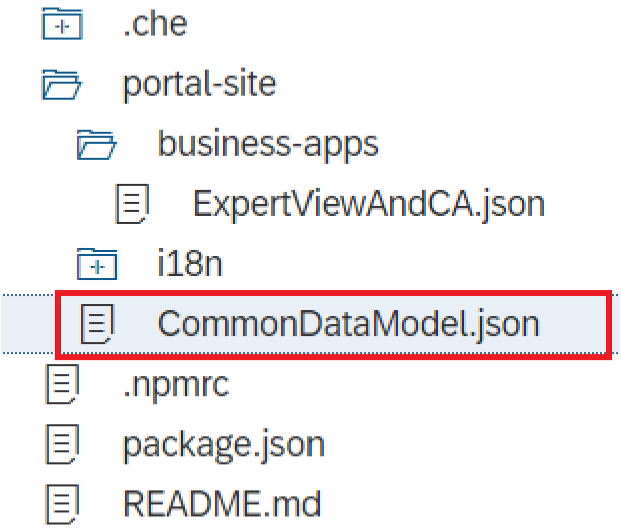
  ```
  "showAdditionalAttributes":{
      "value":{
          "value":"true",
          "format":"plain"
      }
  },
  ```

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.

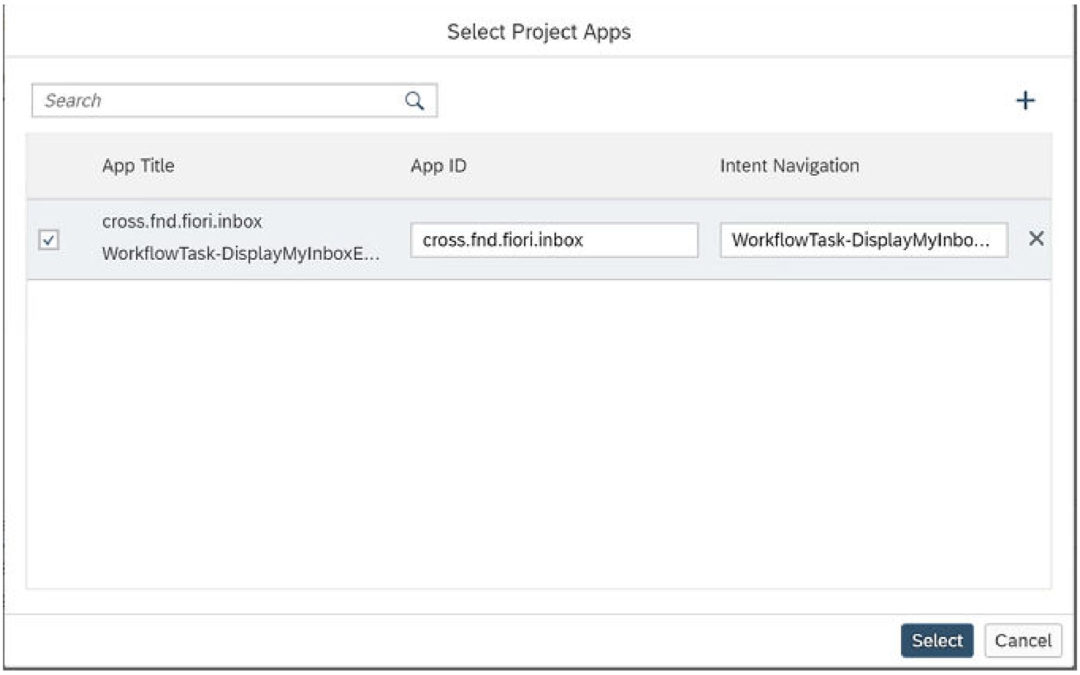


.che
portal-site
business-apps
ExpertViewAndCA.json
i18n
CommonDataModel.json
.npmrc
package.json
README.md

      - 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`



Select Project Apps

	App Title	App ID	Intent Navigation	
☑	cross.fnd.fiori.inbox WorkflowTask-DisplayMyInboxE...	cross.fnd.fiori.inbox	WorkflowTask-DisplayMyInbo...	✕

- 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 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. 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

> ⓘ 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 215].

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 209].

- You’ve created a start form. See Create Your Form [page 137].

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`**

Parameter	Sample Value
`apptitle`	**`Request Approval`**
`formtitle`	**`Create Approval Request`**

> ⓘ 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 Build Work Zone, standard edition and its SAP Fiori launchpad functionalities.

Prerequisites

- You have subscribed to the Launchpad application and configured the necessary roles for your user in your subaccount. See Initial Setup in the SAP Build Work Zone, standard edition documentation.
- You have a workflow capability instance running. See Create a Service Instance of the Workflow Capability Using the Cockpit [page 191].
- You have an SAP Fiori launchpad site. See Create a Site.
- You have created a custom start UI. See Creating a Custom Start UI [page 126].

Context

> ⓘ 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 129].

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 capability, choose ▶ *Content Providers* ▸ *HTML 5 Apps* ◀ and then choose *Refresh* (↻).
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 to your added HTML5 application item.
    b. On the screen that opens, choose *Create a Local Copy*.
    c. To change 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 change 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, and 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 Build Work Zone, standard edition 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 Workflow Capability Data

Export workflow data to provide access to the business data stored within the workflow. For example, you can use this data in an audit .

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 capability.

You can export the following types of data from the workflow capability:

- Design time or modeling artifacts from the workflow editor.
  For more information, see Transport Workflows Between Accounts [page 80] in the SAP BTP, Cloud Foundry environment.
- Runtime artifacts from the workflow capability using the workflow capability 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 workflow capability REST APIs See Using Workflow APIs [page 156].

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 162].

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 of the deployed versions.

- 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 maximum 64 characters long.
substitutedUser	String	The name of the user who is substituted. The user ID is maximum 255 characters long.
substitute	String	The name of the user who substitutes 'substitutedUser'. The user ID is maximum 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.
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 Capability

Administrators of the SAP Business Technology Platform (SAP BTP) account can disable the workflow capability.

Context

> ⚠ Caution
>
> When you delete the last workflow capability instance, all data from your subaccount is deleted.

Procedure

Delete the workflow capability instances.

For more information, see Deleting Service Instances.

4.2 Using the Workflow Capability

This workflow user guide is for end users and key users.

Related Information

Managing Workflows Using the Monitor Workflows App [page 278]

4.2.1 Working with Tasks in My Inbox

You can process workflow service tasks in the My Inbox application, which runs on 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 209].

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 all tasks that are assigned to you.
- Claim tasks.

> ⓘ 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*.

- 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 in the *Expert View* is limited to the first 1000 entries that match the filter. In the *Filtered by* header, you can hover over the title to see information about all of the filters applied on the task list.
- 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.

> ⓘ 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.

> ⓘ 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 121], appear at the bottom of the screen. When you select one of the buttons, a *Custom Action Dialog* opens. You can add a Decision Note. The *Decision Note* field is optional, unless it is marked with an asterisk (*). If the field should be filled in, the *Decision Option* button is active only if this requirement is fulfilled. Otherwise, you can submit your decision without adding a Decision Note.

Rate Limits

To ensure optimal operation of the service, workflow capability 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 262]

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:

- View details about the workflow for a selected task and events, relevant to it chronologically.

  > ⓘ Note
  >
  > The workflow capability 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.
- Sort, group, and filter tasks by *Status*, *Created On*, *Task Type* and *Due On*.
  You can filter the tasks by the following *Due On* categories:
  - *Overdue*- Returns all tasks where the due date is in the past.
  - *Within a Week*- Returns all tasks with a due date in the next 7 calendar days.
  - *Within a Month*- Returns all tasks with a due date in the next 30 calendar days.
  - *All*

- View information related to the due date (for example, *Overdue*, *Within a Week*) if the task has a deadline for processing.
- View workflow-related information.
- Process tasks using custom actions.
  For more information, see Add Task Completion Buttons to My Inbox [page 121].

Expert View

The *Expert* view is a tabular representation of the standard attributes of your tasks in My Inbox.

It allows you to:

- View a 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.

> ⓘ 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 itwith other users.

> ⓘ Note
>
> The personalizations 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.

> ⓘ 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].

> ⓘ 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, in a chronological order.

Claim

To reserve a task for processing, choose *Claim*.

> ⓘ 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.

> ⓘ 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 Standard Expert View 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.

> ⓘ 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*, or *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, choose *Personalize*.

> ⓘ 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.

4.2.1.2.2 Working with Workflow Intelligence

Workflow Intelligence provides confidence level, which allows process participants to make more accurate decisions faster.

Prerequisites

- Workflow Intelligence is enabled, as described in *Confidence Level column in the Expert View* in Configure My Inbox App [page 211].
- Workflow intelligence is available for your region. For more information, see Conventions, Restrictions, and Limits.

Working with Confidence Level

Confidence Level is the calculated value that represents the system's recommendation about the probability of approving a given task.

Using *Confidence Level*, substitutes can take informed decisions with confidence, based on the Workflow Intelligence recommendation.

The *Expert View* in My Inbox displays the *Confidence Level* as a column, and shows the calculated value of the probability for task approval. For more information, see Intelligence Within a Workflow – Confidence Levels for Task Instances [page 171].

You can sort, filter, or group tasks by confidence level.

Sort

Sort by *Confidence Level* in ascending or descending order.

Filter

You can filter by a specific confidence level percentage, or define detailed filter conditions in the *Define Conditions: Confidence Level* dialog (⧉).

Group

The tasks are displayed in the following confidence level groups:

- 80% - 100%
- 60% - 79.9%
- 40% - 59.9%
- 20% - 39.9%
- 0% - 19.9%
- Tasks without confidence level

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. You can create substitution rules for planned and unplanned absences.

At runtime, you can use the respective workflow 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 156].

The audit log data stored for your account is deleted on a regular basis. If you want to retain and use the data after the defined period, you can retrieve it using 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 231] or Create Workflow and My Inbox Tiles on Central Launchpad [page 209].

Planned Substitution

Planned substitution is usually targeted for a scenario where you know the start date and the end date of your absence. Your substitute will then see your tasks displayed directly in their inbox for the period that you defined.

To create a substitution rule for a planned absence:

1. In My Inbox, open the user action menu and select *Manage My Substitutes*.
2. Open the *Planned* tab, and in the footer of the screen choose *Add New Substitute*.
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*.

> ⓘ Note
>
> The *Add Substitute* dialog field is case sensitive. When entering the user ID of the substitute, make sure to use the exact case as displayed in the *Support Information* menu. Otherwise the substitute might not receive tasks during the substitution period.

4. Choose a period for the substitution and choose *Save*.

   > ⓘ Note
   >
   > If you do not select a substitution period and save the rule, it is planned from the day that you created the rule for an undefined period.

   The planned substitution rules are automatically activated on the start date that you selected, and are automatically deactivated on the end date. On the start date of the substitution rule, your substitute will automatically receive the tasks that you have defined in the substitution rule . On the end date of the substitution rule, your substitute will automatically stop receiving the tasks that you have defined in the substitution rule.
   Tasks that have already been claimed by the substitute before 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 an *Active* or *Inactive* status for each substitution rule you have created.

Unplanned Substitution

To create a substitution rule for an unplanned absence:

1. In My Inbox, click the user icon and select *Manage My Substitutes*.
2. Open the *Unplanned* tab.
3. In the footer of the screen, choose *Add New Substitute*.
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*.

   > ⓘ Note
   >
   > The *Add Substitute* dialog field is case sensitive. When entering the user ID of the substitute, make sure to use the exact case as displayed in the *Support Information* menu. Otherwise the substitute might not receive tasks during the substitution period.

5. Choose *Save*.
   Tasks that have already been 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.

> ⓘ 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 defined you as their unplanned substitute.

1. In My Inbox, press the user icon and select *Substitute For*.
   You will see the list of users who have defined you as their unplanned substitute.You will not receiving their tasks until you activate the substitution for that particular user.
2. Click 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.

Restriction for Substitution

Role collections can't be substituted. Only individual users can be substituted.

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's contextual data, for example, Project ID or Project Name. These additional task details allow you to make more informed decisions about tasks when working with My Inbox.

Context

The custom attributes in My Inbox are supported by the *Master-Detail View* and the *Expert View*.

> ⓘ Note
>
> The custom attributes need to be configured in advance so that they are displayed in My Inbox. See Display Custom Attributes in My Inbox [page 41].

Benefits

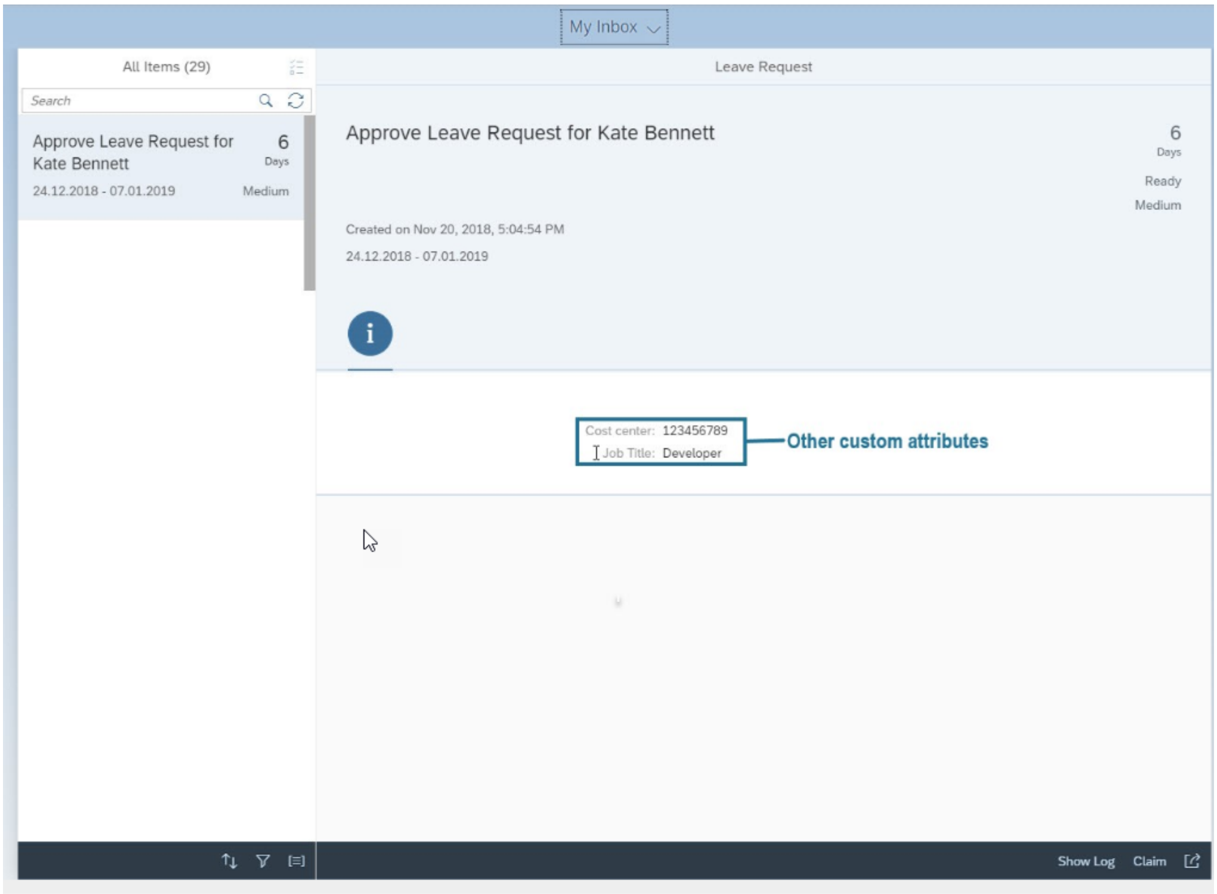
- View additional business data related to tasks to make more informed decisions.
- Quickly find tasks using sorting and filtering in the *Expert View*.

Use

Master-Detail View

The additional information displayed by custom attributes is displayed in specific places on the My Inbox *Master-Detail View*.

My Inbox displays up to 3 KPI indicators in the task list and these are 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, right after the description of the task:



Approve Leave Request for Kate Bennett

Created on Nov 20, 2018, 5:04:54 PM

24.12.2018 - 07.01.2019

Cost center: 123456789

Job Title: Developer

→ Tip

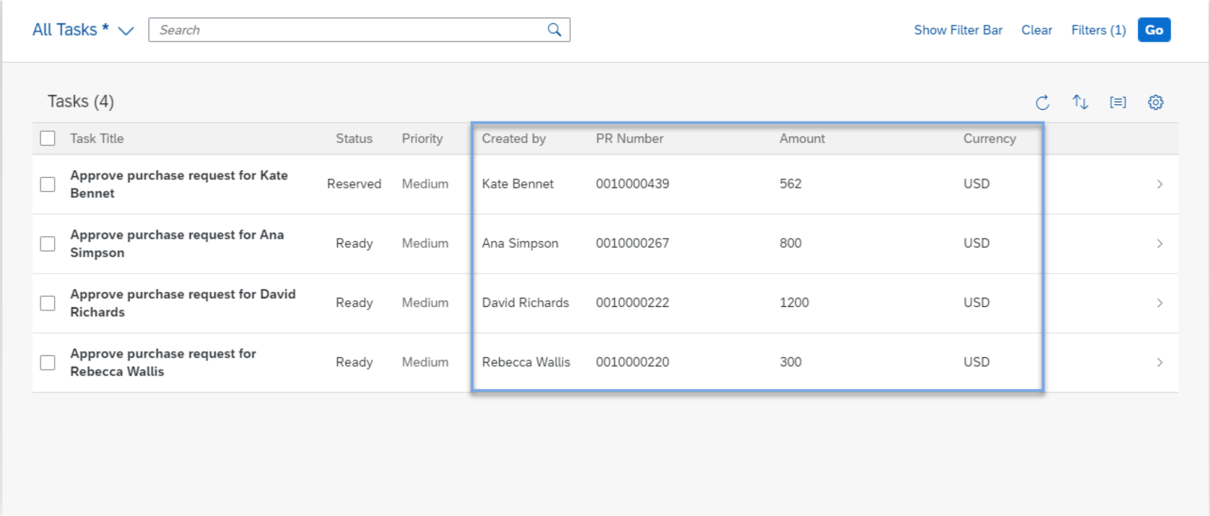
If you are working with a task showing a custom task UI, these custom attributes are not displayed in the task details page of My Inbox. In this case, you can use the respective Workflow API or Task Consumption Model API to search for custom attributes or to find the respective task instances. For more information about these APIs, see Using Workflow APIs [page 156].

ⓘ Note

You cannot sort and filter custom attribute data in the *Master-Details View* of My Inbox. If you want 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.



Tasks (4)

□	Task Title	Status	Priority	Created by	PR Number	Amount	Currency	
□	Approve purchase request for Kate Bennet	Reserved	Medium	Kate Bennet	0010000439	562	USD	>
□	Approve purchase request for Ana Simpson	Ready	Medium	Ana Simpson	0010000267	800	USD	>
□	Approve purchase request for David Richards	Ready	Medium	David Richards	0010000222	1200	USD	>
□	Approve purchase request for Rebecca Wallis	Ready	Medium	Rebecca Wallis	0010000220	300	USD	>

To display the custom attribute data available for tasks, do the following:

> ⓘ Note
>
> In order to see the available custom attributes in the *Expert View*, you first need to filter tasks by *Task Type*.

- Click *Filters* to open the *Filters* dialog.
- Select at least one *Task Type* from the dropdown 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.
- Click *Go* to apply your changes.
- The custom attributes, configured for the chosen Task Types, will be added as columns in the table.

> ⓘ Note
>
> You can change the display of the columns using the *Personalize* button (⚙) for the table. For more information, see Standard Expert View Operations [page 248].

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 216].

Scenario-Specific Tiles in Expert View

The *Expert* view 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 *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 existence of scenario-specific tiles 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 by clicking *Personalize* (⚙) in the table. For more information, see Standard Expert View Operations [page 248].

5 Security

This guide provides an overview of the security-relevant information that applies to the workflow capability.

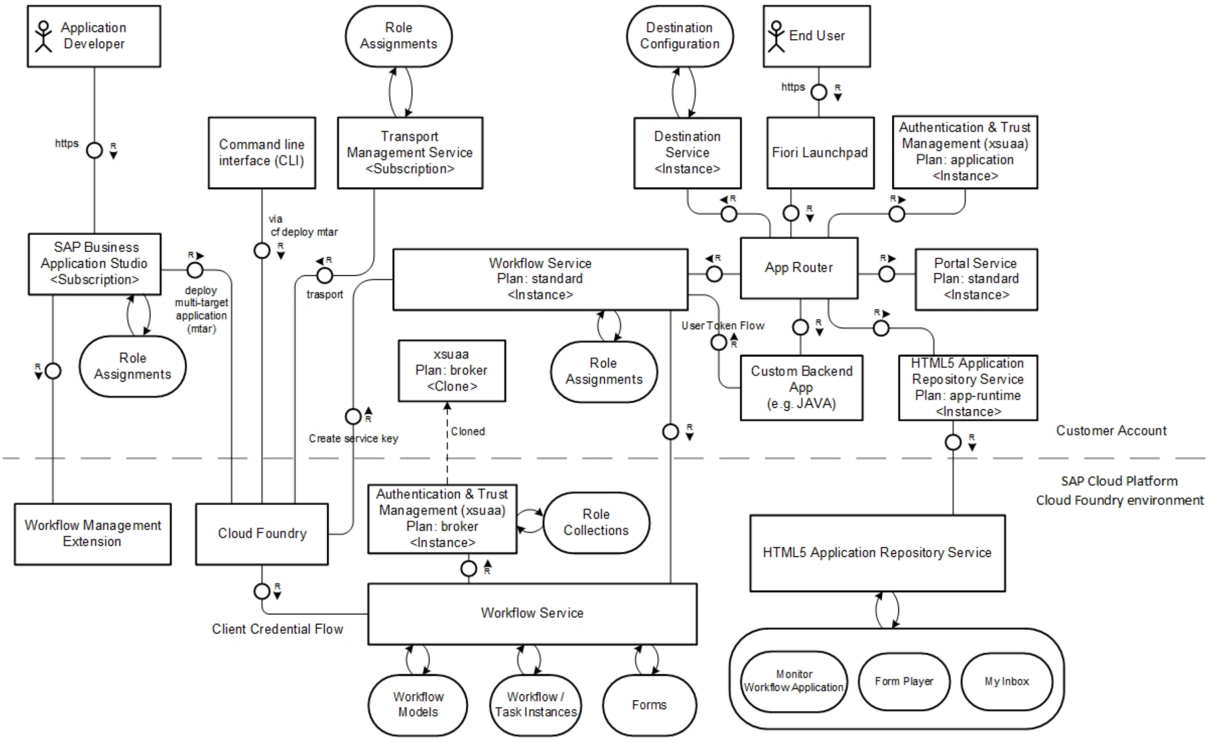
It does not replace the administration guide that is available for productive operation.

Related Information

Security Guide for SAP Business Technology Platform (SAP BTP)

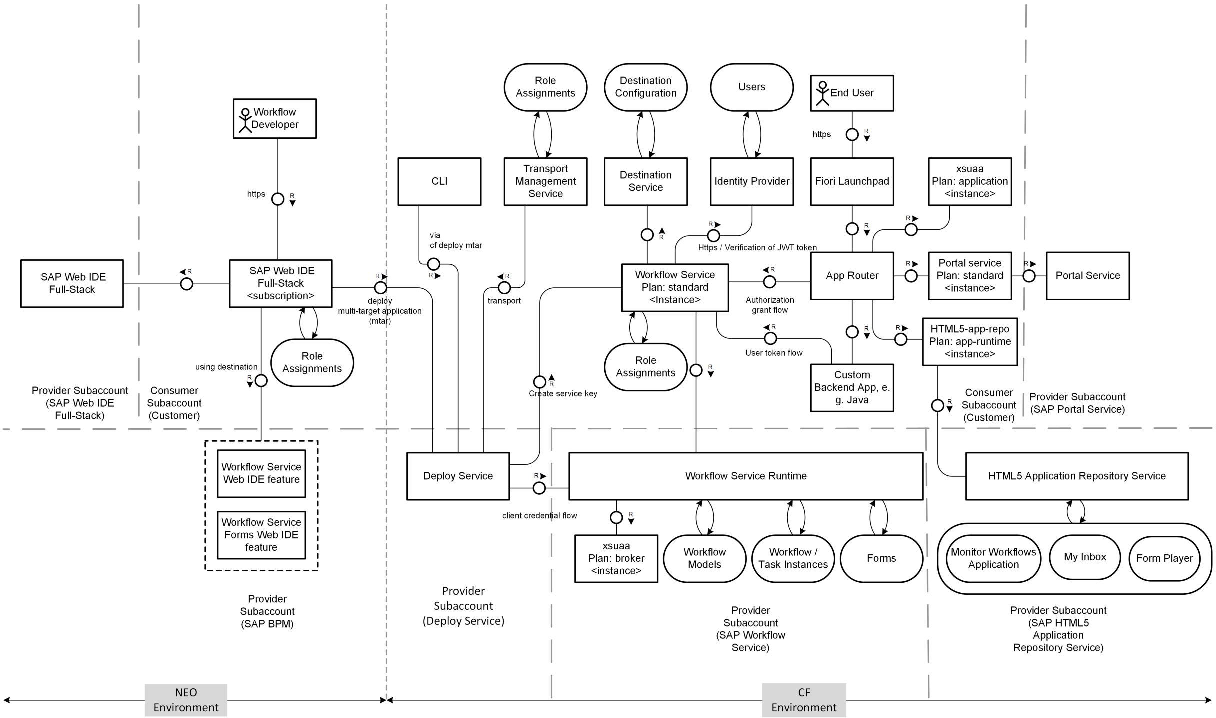
5.1 Architecture

The architecture of the workflow capability comprises several components and subservices.



Application Developer
https
Role Assignments
Command line interface (CLI)
Transport Management Service <Subscription>
Destination Configuration
Destination Service <Instance>
End User
https
Fiori Launchpad
Authentication & Trust Management (xsuaa) Plan: application <Instance>
via cf deploy mtar
SAP Business Application Studio <Subscription>
deploy multi-target application (mtar)
trasport
Workflow Service Plan: standard <Instance>
App Router
Portal Service Plan: standard <Instance>
Role Assignments
xsuaa Plan: broker <Clone>
Role Assignments
User Token Flow
Custom Backend App (e.g. JAVA)
HTML5 Application Repository Service Plan: app-runtime <Instance>
Create service key
Cloned
Customer Account
SAP Cloud Platform Cloud Foundry environment
Workflow Management Extension
Cloud Foundry
Authentication & Trust Management (xsuaa) Plan: broker <Instance>
Role Collections
HTML5 Application Repository Service
Client Credential Flow
Workflow Service
Workflow Models
Workflow / Task Instances
Forms
Monitor Workflow Application
Form Player
My Inbox

Overview of Components and Subservices When Using SAP Business Application Studio


Role Assignments
Destination Configuration
Users
End User
Workflow Developer
https
R
CLI
Transport Management Service
Destination Service
Identity Provider
Fiori Launchpad
xsuaa Plan: application <instance>
via cf deploy mtar
Https / Verification of JWT token
SAP Web IDE Full-Stack
SAP Web IDE Full-Stack <subscription>
deploy multi-target application (mtar)
transport
Workflow Service Plan: standard <Instance>
Authorization grant flow
App Router
Portal service Plan: standard <instance>
Portal Service
User token flow
HTML5-app-repo Plan: app-runtime <instance>
Role Assignments
using destination
Role Assignments
Create service key
Custom Backend App, e. g. Java
Provider Subaccount (SAP Web IDE Full-Stack)
Consumer Subaccount (Customer)
Consumer Subaccount (Customer)
Provider Subaccount (SAP Portal Service)
Workflow Service Web IDE feature
Workflow Service Forms Web IDE feature
Deploy Service
client credential flow
Workflow Service Runtime
HTML5 Application Repository Service
xsuaa Plan: broker <instance>
Workflow Models
Workflow / Task Instances
Forms
Monitor Workflows Application
My Inbox
Form Player
Provider Subaccount (SAP BPM)
Provider Subaccount (Deploy Service)
Provider Subaccount (SAP Workflow Service)
Provider Subaccount (SAP HTML5 Application Repository Service)
NEO Environment
CF Environment

Overview of Components and Subservices When Using SAP Web IDE Full-Stack

The workflow capability 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 capability 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 capability:

- 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 209] and Configure My Inbox App [page 211].
- The workflow capability 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 capability 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 258].

5.2 Identity Provider and Identity Management

For identity management and authentication, the workflow capability relies on the identity provider that is configured in the customer subaccount that owns the respective subscriptions.

All requests handled by the workflow capability subscriptions are authenticated against the identity provider of the customer subaccount and authorized against the role assignments specified in the subscriptions in the customer subaccount. All users who need to interact with the various subservices of the workflow capability 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.

> ⚠ Caution
>
> 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 capability instance. Assign authorities to OAuth clients using OAuth2 client credentials grant.

The workflow capability runtime is one of many SAP Business Technology Platform (SAP BTP) services that you can subscribe to. A service instance of the workflow capability is created when you subscribe to the workflow capability.

Authorization for the workflow capability is provided at the following levels:

- Workflow capability global roles: Users who are assigned to these global roles are then granted the associated permissions for all workflow definitions, instances, and tasks.
- Workflow capability 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 261].
- Instance-specific authorizations: Users who are assigned to these roles are granted permission only for the respective workflow instance. The workflow capability 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. It is also possible to use the script task API. For more information, see Workflow Definition versus Workflow Instance [page 5] and Get and Set Instance-Specific Roles [page 60].

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].

> ⓘ Note
>
> The scope of data separation is on the SAP BTP subaccount level. Multiple SAP BTP Cloud Foundry dev spaces with different authorization configurations may be used, however they all access the same data.

Roles for Accessing the Workflow Capability 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 • Permission 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)

Role	Description
`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 ⓘ Note This role is a prerequisite for working with instance-specific permissions.

`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)	ⓘ 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

Role	Description
`WorkflowBusinessExpert` (global role)	- Permission to work with process variants.

* Only for global roles

Available Platform Roles for Using the Workflow Capability Runtime

Role	Description
`Space Developer`	- Permission to deploy a workflow project

5.3.1 Technical Authentication

Access the workflow capability 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 capability 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 196]

5.4 Destinations

Subservices communicate using predefined destinations in a customer subaccount, for example, when My Inbox or the *Manage Workflows* application communicates with the workflow runtime.

Predefined destinations are generated and configured when you enable the workflow capability in a customer subaccount. For more information, see Principal Propagation for User Interfaces [page 262] below.

The workflow runtime communicates with other services according to the workflow definitions.

- The workflow runtime uses destinations of type `Mail` to communicate with mail servers.
  For more information, see Configure the Workflow Capability Mail Destination [page 199] and Configure Mail Tasks [page 62].
- To communicate with other services, the workflow runtime uses destinations of type `HTTP`.
  For more information, see Destination Configuration for Service Task [page 262] below and Configure Service Tasks [page 46]. For authentication and authorization purposes, other referenced destinations might also be used, for example, OAuth2 authorization endpoints.

Secure Communication Protocols and Cipher Suites

We recommend that the services or servers maintained in the destinations support at least the TLS 1.2 secure communication protocol. The list of supported cipher suites might vary between private and public cloud deployments and/or data centers.

Note that 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 capability runtime. This allows the posting of all requests that are sent to the workflow capability runtime on behalf of the user (who initiated the request from the user interface).

Principal propagation is automatically enabled when you enable the workflow capability in a customer subaccount.

Destination Configuration for Service Tasks

The workflow capability 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 capability.

You can call services using the following authentication types:

- *Basic Authentication*: Select *Basic Authentication* as the authentication type in the destination.
- *OAuth2-based Authentication*
  For the OAuth-based authentication types, all token-related properties that are documented are supported, for example, `tokenServiceURL` headers or the scope property. All properties that are mere conventions are not supported, for example, the URL prefix.
  - *OAuth2ClientCredentials* flow: Select *OAuth2ClientCredentials* as the authentication type in the destination. For more information, 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 265] and Configuring Principal Propagation for Service Tasks [page 198].
  - *OAuth2Password*: For calls to OAuth-protected services supporting the OAuth password grant flow. If the OAuth2 Client Credentials flow is not available, we recommend that you use this authentication type to authenticate based on technical users. Do not use this authentication type for human users. Also, there is no support for resolving two-factor-authentication mechanisms. For more information, see OAuth Password Authentication.
  - *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 198].
- *ClientCertificateAuthentication*: For calls to services that authenticate the request using client certificates based on the X.509 standard. For more information, see Use Destination Certificates. Note that only certificates maintained at the destination are supported. Ensure that the *Use client provided certificates* check box is not selected. See also the notes below about further limitations that apply.
- *No Authentication*: If the service you want to call doesn't require any authentication, select *No Authentication* as the authentication type in the destination.

In addition to the authentication type, the following destination features are supported in the workflow capability:

- Server authentication (verification): JDK default and custom truststores (for HTTP destinations in general), a key store for type `ClientCertificateAuthentication`.
- 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.

> ⓘ 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.

To configure destinations, use the standard SAP Business Technology Platform (SAP BTP) mechanisms in the SAP BTP cockpit. For more information, see Managing Destinations.

> ⓘ Note
>
> Certain destination configurations are not supported, even though they can be configured in the destination editor. For clarity, the following unsupported configurations are explicitly mentioned:
>
> - The additional `sap-client` property is not supported, although certain other BTP services support it. For information about how to specify the ABAP client in a destination used by workflow, see the Guided Answer.
> - The `PrincipalPropagation` authentication type is not supported. While similar in name to the respective service task feature (see Configure Service Tasks [page 46]), it is an authentication type related to SAP Connectivity Service for on-premise connectivity. Propagating principals is only possible with the `OAuth2SAMLBearerAssertion` and `OAuth2UserTokenExchange` authentication types.

> ⓘ 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.
>
> The workflow capability 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 capability runtime must temporarily access this data when executing a workflow instance. This data isn't persisted within the workflow capability 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

- You have modeled 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

> ⓘ 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 262].

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 to 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.

> ⓘ 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 capability 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
>
> The workflow capability 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 156] and Export Workflow Capability Data [page 240].

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 capability is deleted.

For audit needs, the workflow capability offers an export feature. For more information, see Export Workflow Capability Data [page 240].

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 workflow capability 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, the workflow capability may write logs into the audit log handled by the platform itself.

> ⓘ Note
>
> The workflow capability 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 capability 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 267].

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 already defined when the purpose starts.
Consent	The action of the data subject confirming that the usage of their 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, the 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 for 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 that 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 capability 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*.

Security Events Written in Audit Logs

Event Grouping	What events are logged	How to identify related log events
Tenant events	Create new tenant - `about-to-create-account-data`	`SecurityEventAuditMessage` sub-category `data-creation` Creation of data for <account> started by <user>
	Create new tenant - `create-account-data-done`	`SecurityEventAuditMessage` sub-category `data-creation` Creation of data for <account> completed
	Create new tenant - `create-account-data-failed`	`SecurityEventAuditMessage` sub-category `data-creation` Creation of data for <account> failed. Reason for failing the creation is <Reason>
	Delete existing tenant - `requested-to-delete-account-data`	`SecurityEventAuditMessage` sub-category `data-termination` Deletion (reason: <reason>) of data for <account> requested by <user>
	Delete existing tenant - `about-to-delete-account-data`	`SecurityEventAuditMessage` sub-category `data-termination` Deletion (reason: <reason>) of data for <account> started by <user>
	Delete existing tenant - `delete-account-data-done`	`SecurityEventAuditMessage` sub-category `data-termination` Deletion (reason: <reason>) of data for <account> completed
	Delete existing tenant - `delete-account-data-failed`	`SecurityEventAuditMessage` sub-category `data-termination` Deletion (reason: <reason>) of data for <account> failed. Reason for failing the deletion is <Reason>
	Purge all workflow data of a tenant - `purge-triggered`	`SecurityEventAuditMessage` sub-category `workflow-apptenant-purge` User &purgingUser triggered purging of all data (containing &countWorkflowDefinitions WorkflowDefinitions) in tenant &tenant of appTenant &appTenant via channel &channel. In case forms purge is 'enabled': User &purgingUser triggered purging of all data (containing &countWorkflowDefinitions WorkflowDefinitions and all FormDefinitions) in tenant &tenant of appTenant &appTenant via channel &channel.
	Purge all workflow data of a tenant - `about-to-purge`	`SecurityEventAuditMessage` sub-category `workflow-apptenant-purge` Job triggered by user &purgingUser is now starting to purge all data in tenant &tenant of appTenant &appTenant via channel &channel.

Event Grouping	What events are logged	How to identify related log events
	Purge all workflow data of a tenant - `purge-done`	`SecurityEventAuditMessage` sub-category `workflow-apptenant-purge` Job triggered by user &purgingUser is completed with purging all data in tenant &tenant of appTenant &appTenant via channel &channel.
	Purge all workflow data of a tenant - `purge-failed`	`SecurityEventAuditMessage` sub-category `workflow-apptenant-purge` General failure: Job triggered by user &purgingUser has failed while purging all data in tenant &tenant of appTenant &appTenant via channel &channel. Failure during workflow artefacts purge: Job triggered by user &purgingUser has failed while purging all data in tenant &tenant of appTenant &appTenant via channel &channel. &countWorkflowDefinitions Workflow Definitions not purged. Failure during forms artefacts purge: Job triggered by user &purgingUser has failed while purging all forms data in tenant &tenant of appTenant &appTenant via channel &channel.
	Export all workflow data from a tenant - `about_to_read`	`SecurityEventAuditMessage` sub-category `data-export` User &userId is about to export all data of tenant &tenantDescription.
	Export all workflow data from a tenant - `read`	`SecurityEventAuditMessage` sub-category `data-export` User &userId finished exporting all data of tenant &tenantDescription.
	Export all workflow data from a tenant - `read-failed`	`SecurityEventAuditMessage` sub-category `data-export` Export of all data of tenant &tenantDescription for user &user failed during zip streaming. [Log-ID: &logId]
Instance events	Create a new service instance - `about-to-create-service-instance`	`SecurityEventAuditMessage` sub-category `data-creation` Creation of service instance with ID &serviceInstanceId in space &spaceGuid of organization &organizationGuid with parameters &parameters started by &userId
	Create a new service instance - `create-service-instance-done`	`SecurityEventAuditMessage` sub-category `data-creation` Creation of service instance with ID &serviceInstanceId in space &spaceGuid of organization &organizationGuid with parameters &parameters completed
	Create a new service instance - `create-service-instance-failed`	`SecurityEventAuditMessage` sub-category `data-creation` Creation of service instance with ID &serviceInstanceId in space &spaceGuid of organization &organizationGuid with parameters &parameters failed with error message: <error>

Event Grouping	What events are logged	How to identify related log events
	Delete an existing service instance - `requested-to-delete-service-instance`	`SecurityEventAuditMessage` sub-category `data-termination` Deletion of service instance with ID &serviceInstanceId requested by &userId
	Delete an existing service instance - `about-to-delete-service-instance`	`SecurityEventAuditMessage` sub-category `data-termination` Deletion of service instance with ID &serviceInstanceId started by &userId
	Delete an existing service instance - `delete-service-instance-done`	`SecurityEventAuditMessage` sub-category `data-termination` Deletion of service instance with ID &serviceInstanceId completed
	Delete an existing service instance - `delete-service-instance-failed`	`SecurityEventAuditMessage` sub-category `data-termination` Deletion of service instance with ID &serviceInstanceId failed with error message: <error>
	Update an existing service instance - `about-to-update-service-instance`	`SecurityEventAuditMessage` sub-category `data-modification` Update of service instance with ID &serviceInstanceId with requested changes &requestedChanges started by &userId
	Update an existing service instance - `update-service-instance-done`	`SecurityEventAuditMessage` sub-category `data-modification` Update of service instance with ID &serviceInstanceId with updated parameters &parameters completed
	Update an existing service instance - `update-service-instance-failed`	`SecurityEventAuditMessage` sub-category `data-modification` Update of service instance with ID &serviceInstanceId with updated parameters &parameters failed with error message: <error>
	Create a new subscription - `create-subscription-started`	`SecurityEventAuditMessage` sub-category `data-creation` Subscription for tenant &tenantId to application &appName started by &userId
	Create a new subscription - `create-subscription-completed`	`SecurityEventAuditMessage` sub-category `data-creation` Subscription for tenant &tenantId to application &appName completed
	Create a new subscription - `create-subscription-failed`	`SecurityEventAuditMessage` sub-category `data-creation` Subscription for tenant &tenantId to application &appName failed with error message: <error>

Event Grouping	What events are logged	How to identify related log events
	Delete an existing subscription - `delete-subscription-started`	`SecurityEventAuditMessage` sub-category`data-termination` Unsubscription of tenant &tenantId from application &appName started by &userId
	Delete an existing subscription - `delete-subscription-completed`	`SecurityEventAuditMessage` sub-category`data-termination` Unsubscription of tenant &tenantId from application &appName completed
	Delete an existing subscription - `delete-subscription-failed`	`SecurityEventAuditMessage` sub-category`data-termination` Unsubscription for tenant &tenantId from application &appName failed with error message: <error>
	Patched a workflow instance - `context-patched`	`SecurityEventAuditMessage` Context of workflow instance '&workflowInstanceId' was patched by user '&userId'.
	Context of a workflow instance was overwritten - `context-overwritten`	`SecurityEventAuditMessage` Context of workflow instance '&workflowInstanceId' was overwritten by user '&userId'.
Deployment events	Deploy a workflow definition - `deploy-done`	`SecurityEventAuditMessage` sub-category`workflow-definition-deployment` User '&userId' deployed version &version of workflow definition '&definitionId' in tenant '&tenantId' and app tenant '&reDeployAppTenant', via tenant '&appTenant'.
	Undeploy a workflow definition - `undeploy-scheduled`	`SecurityEventAuditMessage` sub-category `workflow-definition-undeployment` User '&userId' triggered the undeployment of the workflow definition '&definitionId' in tenant '&tenantId' and app tenant '&appTenant'.
	Undeploy a workflow definition - `about-to-undeploy`	`SecurityEventAuditMessage` sub-category `workflow-definition-undeployment` Job for undeploying workflow definition '&definitionId' in tenant '&tenantId' and app tenant '&appTenant' has been started.
	Undeploy a workflow definition - `undeploy-done`	`SecurityEventAuditMessage` sub-category `workflow-definition-undeployment` Job for undeploying workflow definition '&definitionId' in tenant '&tenantId' and app tenant '&appTenant' has been completed.

Event Grouping	What events are logged	How to identify related log events
	Undeploy a workflow definition - `undeploy-failed`	`SecurityEventAuditMessage` sub-category `workflow-definition-undeployment` Job for undeploying workflow definition '&definitionId' in tenant '&tenantId' and app tenant '&appTenant' has been failed.
	Deploy a form definition - `about-to-deploy`	`SecurityEventAuditMessage` sub-category `form-definition-deployment` Deploying form definition '&definitionId' in tenant '&tenant' and app tenant '&appTenant' by user '&user' has been started.
	Deploy a form definition - `deploy-done`	`SecurityEventAuditMessage` sub-category `form-definition-deployment` Deploying form definition '&definitionId' in tenant '&tenant' and app tenant '&appTenant' by user '&user' has been completed.
	Deploy a form definition - `deploy-failed`	`SecurityEventAuditMessage` sub-category `form-definition-deployment` Deploying form definition '&definitionId' in tenant '&tenant' and app tenant '&appTenant' by user '&user' has been failed.
	Undeploy a form definition - `about-to-undeploy`	`SecurityEventAuditMessage` sub-category `form-definition-undeployment` Undeploying form definition '&definitionId' in tenant '&tenant' and app tenant '&appTenant' by user '&user' has been started.
	Undeploy a form definition - `undeploy-done`	`SecurityEventAuditMessage` sub-category `form-definition-undeployment` Undeploying form definition '&definitionId' in tenant '&tenant' and app tenant '&appTenant' by user '&user' has been completed.
	Undeploy a form definition - `undeploy-failed`	`SecurityEventAuditMessage` sub-category `form-definition-undeployment` Undeploying form definition '&definitionId' in tenant '&tenant' and app tenant '&appTenant' by user '&user' has been failed.
Workflow Instances events	Delete a workflow instance - `aboutToDelete`	`SecurityEventAuditMessage` sub-category `workflow-instances` Deletion of workflow instances was requested in app tenant '&appTenant' with sharedContentIncluded=&sharedContentIncluded: <instanceIds>
	Delete a workflow instance - `delete`	`SecurityEventAuditMessage` sub-category `workflow-instances` The following workflow instances were deleted from app tenant '&appTenant' with sharedContentIncluded=&sharedContentIncluded: <instanceIds>

Event Grouping	What events are logged	How to identify related log events
Event Subscriptions events	Create, update, and delete event subscriptions - `create`	`ConfigurationChangeAuditMessage` sub-category `event-subscription`
	Create, update, and delete event subscriptions - `createFailed`	`ConfigurationChangeAuditMessage` sub-category `event-subscription`
	Create, update, and delete event subscriptions - `update`	`ConfigurationChangeAuditMessage` sub-category `event-subscription`
	Create, update, and delete event subscriptions - `updateFailed`	`ConfigurationChangeAuditMessage` sub-category `event-subscription`
	Create, update, and delete event subscriptions - `delete`	`ConfigurationChangeAuditMessage` sub-category `event-subscription`
	Create, update, and delete event subscriptions - `deleteFailed`	`ConfigurationChangeAuditMessage` sub-category `event-subscription`
Security events	Update instance-specific role assignments of a workflow instance	`ConfigurationChangeAuditMessage` sub-category `workflow-instances`
Substitution events	Create, update, and delete event substitution rules - `create`	`SecurityEventAuditMessage` sub-category `substitution-rule`
	Create, update, and delete event substitution rules - `createFailed`	`SecurityEventAuditMessage` sub-category `substitution-rule`
	Create, update, and delete event substitution rules - `delete`	`SecurityEventAuditMessage` sub-category `substitution-rule`
	Create, update, and delete event substitution rules - `deleteFailed`	`SecurityEventAuditMessage` sub-category `substitution-rule`

Related Information

Audit Logging in the Cloud Foundry Environment

6 Monitoring and Troubleshooting

When working with the workflow capability, 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 Capability.

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:

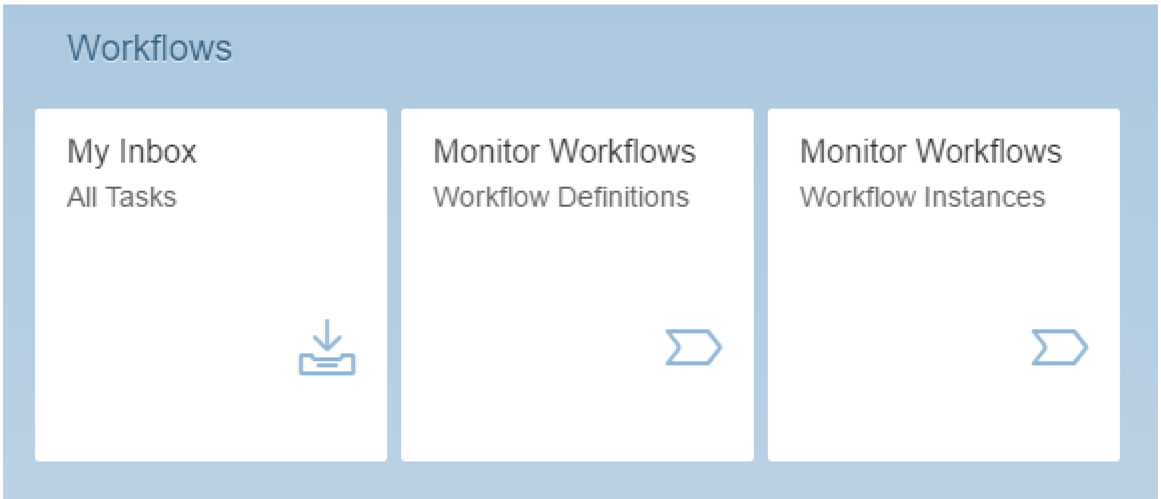
- Landscape information (Canary, EU10, US10)
- The URL of the page where the incident or error occurs
- The steps or clicks used to replicate the error
- Screenshots, videos, or the code entered

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 209].



ⓘ 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

Workflow Instances [page 279]
Managing Task Instances [page 280]
Workflow Definitions [page 281]
Deep Linking in Monitor Workflows App [page 283]

6.1.1 Workflow Instances

The Workflow Instances view shows a list of all workflow instances and offers actions to work on the instances.

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.

  > ⓘ Note

- 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 the subflow instances that are started by a workflow instance. Choose *Show Subflow Instances*.
- View tasks of a workflow instance. Choose *Show Tasks*.
-
-
- 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.
-
- Resume a suspended workflow instance. Choose *Resume*. This operation also retries failed steps.
- Navigate to the workflow definition of an instance.
- Navigate to the parent workflow instance for a subflow.
- Navigate to the root instance of a workflow or subflow.

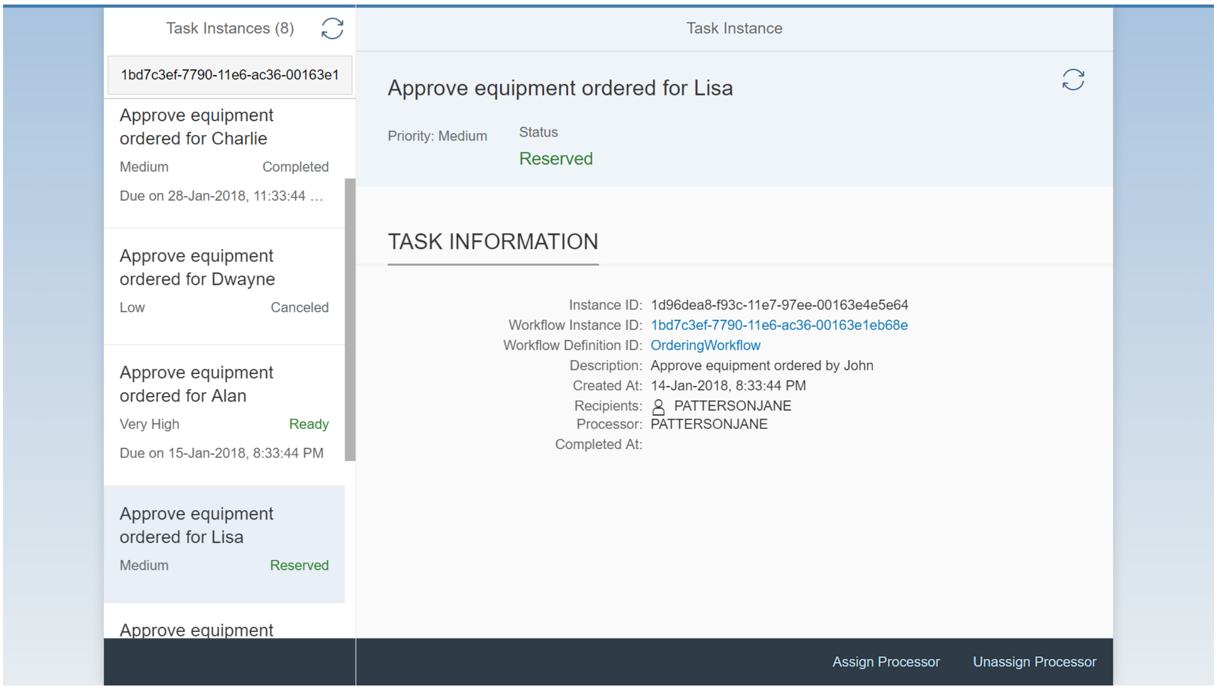
  > ⓘ Note
  >
  > Root instances are main workflow instances that are started by an application or a user. Main workflow instances in turn start subflow instances.

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 209].



Task Instances (8)

1bd7c3ef-7790-11e6-ac36-00163e1

Approve equipment ordered for Charlie
Medium Completed
Due on 28-Jan-2018, 11:33:44 ...

Approve equipment ordered for Dwayne
Low Canceled

Approve equipment ordered for Alan
Very High Ready
Due on 15-Jan-2018, 8:33:44 PM

Approve equipment ordered for Lisa
Medium Reserved

Approve equipment

Task Instance

Approve equipment ordered for Lisa

Priority: Medium

Status
Reserved

TASK INFORMATION

Instance ID: 1d96dea8-f93c-11e7-97ee-00163e4e5e64
Workflow Instance ID: 1bd7c3ef-7790-11e6-ac36-00163e1eb68e
Workflow Definition ID: OrderingWorkflow
Description: Approve equipment ordered by John
Created At: 14-Jan-2018, 8:33:44 PM
Recipients: PATTERSONJANE
Processor: PATTERSONJANE
Completed At:

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 278]
Deep Linking in Monitor Workflows App [page 283]

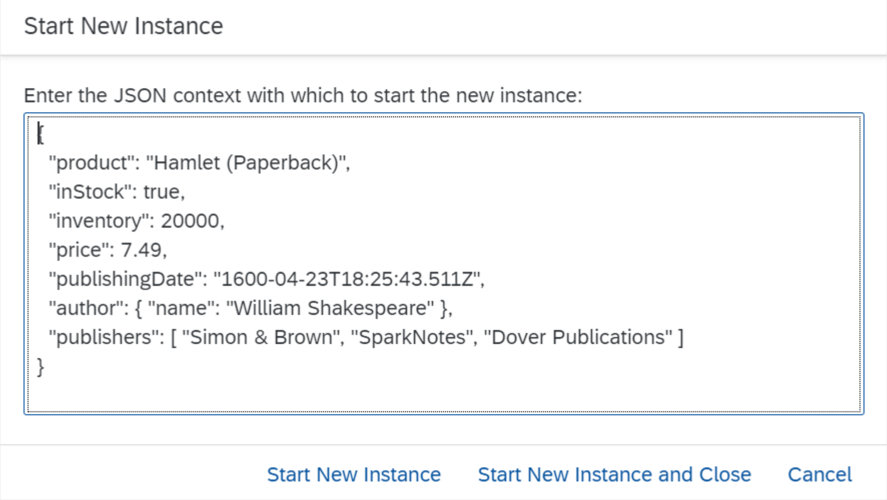
6.1.3 Workflow Definitions

The Workflow Definitions view shows a list of deployed workflow definitions.

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.

  Example

  Start New Instance

  Enter the JSON context with which to start the new instance:

  ```
  {
    "product": "Hamlet (Paperback)",
    "inStock": true,
    "inventory": 20000,
    "price": 7.49,
    "publishingDate": "1600-04-23T18:25:43.511Z",
    "author": { "name": "William Shakespeare" },
    "publishers": [ "Simon & Brown", "SparkNotes", "Dover Publications" ]
  }
  ```

  The JSON structure contains the content to be passed to the workflow context. In contrast to the workflow capability 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 capability API documentation in Uisng Workflow APIs.
- 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.

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://<consumer_account>.launchpad.<landscape_host>/site?siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances`
- To access a particular workflow instance, use the following URL format:
  `https://<consumer_account>.launchpad.<landscape_host>/site?siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances/<workflow_instance_id>`

Workflow Definition

- To access the list of workflow definitions, use the following URL format:
  `https://<consumer_account>.launchpad.<landscape_host>/site?siteId=<site_id>#bpmworkflowmonitor-DisplayDefinitions&/workflowDefinitions`
- To access a particular workflow definition, use the following URL format:
  `https://<consumer_account>.launchpad.<landscape_host>/site?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://<consumer_account>.launchpad.<landscape_host>/site?siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances/<workflow_instance_id>/taskInstances`
- To access a particular task instance, use the following URL format:
  `https://<consumer_account>.launchpad.<landscape_host>/site?siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances/<workflow_instance_id>/taskInstances/<task_instance_id>`

> Note
>
> If you are using the default site, then `site_id` in the URL is not mandatory.

Related Information

Workflow Instances [page 279]
Managing Task Instances [page 280]
Workflow Definitions [page 281]

6.2 Traceability during Transport of Workflow Modules

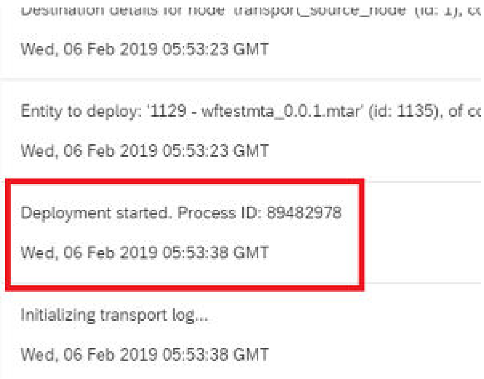
You can use the transport and deploy logs to trace the details while transporting an MTA archive between two subaccounts.

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.



```
Destination details for node 'transport_source_node' (id: 1), c
Wed, 06 Feb 2019 05:53:23 GMT

Entity to deploy: '1129 - wftestmta_0.0.1.mtar' (id: 1135), of c
Wed, 06 Feb 2019 05:53:23 GMT

Deployment started. Process ID: 89482978
Wed, 06 Feb 2019 05:53:38 GMT

Initializing transport log...
Wed, 06 Feb 2019 05:53:38 GMT
```

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.

   > ⓘ 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 an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

THE BEST RUN SAP