## Content

1. **What's New for Workflow Service.** ................................................................. 4

2. **Feature Scope Description.** ........................................................................... 15
   - 2.1 About This Document. ............................................................................. 15
   - 2.2 Feature Scope Description. ..................................................................... 15
   - 2.3 Browser Support. .................................................................................... 16
   - 2.4 Availability. ............................................................................................ 16

3. **Administration.** ........................................................................................... 18
   - 3.1 Introduction. ......................................................................................... 18
   - 3.2 Configure the Workflow Service. .......................................................... 18
     - Configure SAP Fiori Launchpad Objects. ............................................... 20
   - 3.3 Manage Workflows Using the Monitor Workflows App. ....................... 22
   - 3.4 Export Workflow Service Data. ............................................................ 26

4. **Develop.** ......................................................................................................... 28
   - 4.1 Introduction. .......................................................................................... 28
   - 4.2 General. .................................................................................................. 28
     - Workflow Definition versus Workflow Instance. .................................... 31
     - Conventions. ............................................................................................ 31
   - 4.3 Modeling a Workflow. ........................................................................... 32
     - Enable the Workflow Editor in SAP Web IDE. ......................................... 33
     - Importing Workflow Models. .................................................................. 33
     - Create a New Workflow Project. .............................................................. 35
     - Define Workflows. .................................................................................. 38
     - Transport Workflows between Accounts. ............................................. 60
     - Deploy Workflows. .................................................................................. 60
     - Accelerated Modeling with Speed Buttons. ............................................ 61
     - Expressions. ............................................................................................ 62
   - 4.4 Developing a User Interface for a Workflow. .......................................... 65
     - Custom Task UI. ..................................................................................... 66
     - Creating Start UI. ................................................................................. 76
     - Data Propagation from My Inbox to Task Application. ........................... 81
   - 4.5 Using Workflow APIs. ............................................................................. 85
     - Access Workflow APIs Using OAuth 2.0 Authentication (Client Credentials). 86
     - Access Workflow APIs Using OAuth 2.0 Authentication (Authorization Grant). 87
     - Determine the Service Host. .................................................................. 88
     - Modifying the Context of a Workflow Instance. .................................... 89
5 User Guide. ................................................................. 95
  5.1 Introduction. ............................................................. 95
  5.2 Working with Tasks in My Inbox. ................................. 95
    Accessing the My Inbox Application. ............................. 96

6 Security Guide. ............................................................. 97
  6.1 Architecture. ............................................................ 97
  6.2 Identity Provider and Identity Management. .................. 98
  6.3 Authorization Configuration. ....................................... 99
  6.4 Destinations. ........................................................... 101
  6.5 Data Protection and Data Privacy. ................................. 102

7 Troubleshooting. ........................................................... 104
  7.1 End Users Can’t Open SAP Fiori Launchpad Tiles. ........... 104
  7.2 Error When Clicking “Go to Service” on Portal Tile. ........ 106
  7.3 HTTP Status 403: User Doesn’t Have Sufficient Privileges. 107
  7.4 Tasks Not Appearing. .................................................. 107
  7.5 Error During Workflow Deployment in SAP Web IDE. ....... 107
  7.6 Error in JavaScript Files. .............................................. 108
  7.7 No Permissions Granted. .............................................. 108
1 What's New for Workflow Service

The What's New gives you an overview of new and enhanced features and functions of the SAP Cloud Platform Workflow.

28 November 2017 (CW48) - Workflow Service

Table 1:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>With the REST API, the following task properties can be updated:</td>
</tr>
<tr>
<td>• Due date</td>
</tr>
<tr>
<td>• Priority</td>
</tr>
<tr>
<td>• Description</td>
</tr>
<tr>
<td>• Subject</td>
</tr>
<tr>
<td>For more information, see SAP Cloud Platform Workflow API.</td>
</tr>
</tbody>
</table>

Table 2:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>In My Inbox, you can sort by the user who created a task. This information is derived from the initiator of the workflow instance who created the task. It is also available in the workflow service API.</td>
</tr>
<tr>
<td>For more information, see SAP Cloud Platform Workflow API.</td>
</tr>
</tbody>
</table>

21 November 2017 (CW47) - Workflow Service

Table 3:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor user interface enhancements in the editor layout.</td>
</tr>
<tr>
<td>For more information, see Editor Layout [page 37]</td>
</tr>
</tbody>
</table>
13 November 2017 (CW46) - Workflow Service

Table 4:

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can configure the date by when a user task is due. End users can use the due date in My Inbox to sort and filter the tasks.</td>
</tr>
<tr>
<td>For more information, see:</td>
</tr>
<tr>
<td>● Configure User Tasks [page 43]</td>
</tr>
<tr>
<td>● Working with Tasks in My Inbox [page 95]</td>
</tr>
</tbody>
</table>

Table 5:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor user interface changes while configuring the timer events.</td>
</tr>
<tr>
<td>For more information, see:</td>
</tr>
<tr>
<td>● Configure Intermediate Timer Events [page 41]</td>
</tr>
<tr>
<td>● Configure Boundary Timer Events [page 45]</td>
</tr>
</tbody>
</table>

7 November 2017 (CW45) - Workflow Service

Table 6:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the Monitor Workflows app, you can suspend and resume a workflow instance. Moreover, based on customer feedback, the Cancel action was renamed to Terminate.</td>
</tr>
<tr>
<td>For more information, see Manage Workflows Using the Monitor Workflows App [page 22].</td>
</tr>
</tbody>
</table>

Table 7:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the My Inbox, end users can see and filter using the task names that are defined in the editor.</td>
</tr>
<tr>
<td>For more information, see Working with Tasks in My Inbox [page 95].</td>
</tr>
</tbody>
</table>

Table 8:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the My Inbox, users can group tasks by task types.</td>
</tr>
<tr>
<td>For more information, see Working with Tasks in My Inbox [page 95].</td>
</tr>
</tbody>
</table>
17 October 2017 (CW42) - Workflow Service

Table 9:

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clients have the option to also authenticate using OAuth 2.0 when using workflow service APIs.</td>
</tr>
<tr>
<td>For more information, see Access Workflow APIs Using OAuth 2.0 Authentication (Client Credentials) [page 86] and Access Workflow APIs Using OAuth 2.0 Authentication (Authorization Grant) [page 87].</td>
</tr>
</tbody>
</table>

Table 10:

<table>
<thead>
<tr>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can create JavaScript files directly from Script Task Properties.</td>
</tr>
<tr>
<td>For more information, see Configure Script Tasks [page 51].</td>
</tr>
</tbody>
</table>

Table 11:

<table>
<thead>
<tr>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can set the priority while configuring a user task. The priority will be displayed in My Inbox and can also be accessed using the API to filter and sort tasks.</td>
</tr>
<tr>
<td>For more information, see Configure User Tasks [page 43].</td>
</tr>
</tbody>
</table>

Table 12:

<table>
<thead>
<tr>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can configure the user interface properties in a user task using a lookup provided for the SAPUI5 projects in your workspace.</td>
</tr>
<tr>
<td>For more information, see Configure User Tasks [page 43].</td>
</tr>
</tbody>
</table>

10 October 2017 (CW41) - Workflow Service

Table 13:

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can export runtime data related to workflow definitions and workflow instances. You can use this data to address, for example, audit needs.</td>
</tr>
<tr>
<td>For more information, see Export Workflow Service Data [page 26].</td>
</tr>
</tbody>
</table>

Table 14:

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the REST API, task-related execution log entries contain the task instance ID of the respective task.</td>
</tr>
</tbody>
</table>
### 25 September 2017 (CW39) - Workflow Service

**Table 15:**

<table>
<thead>
<tr>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>The standalone workflow editor is deprecated.</td>
</tr>
</tbody>
</table>

**Table 16:**

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can model an intermediate timer event that allows you to pause and resume after a specified interval of time.</td>
</tr>
<tr>
<td>For more information, see Configure Intermediate Timer Events [page 41].</td>
</tr>
</tbody>
</table>

**Table 17:**

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can model a boundary timer event for a user task, when you want an alternative flow to be triggered, if the user task is not completed within the specified time duration.</td>
</tr>
<tr>
<td>For more information, see Configure Boundary Timer Events [page 45].</td>
</tr>
</tbody>
</table>

### 19 September 2017 (CW38) - Workflow Service

**Table 18:**

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>The workflow deployment errors are displayed in the Problems view of SAP Web IDE Full-Stack.</td>
</tr>
<tr>
<td>For more information, see Deploy Workflows [page 60].</td>
</tr>
</tbody>
</table>

**Table 19:**

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can configure the My Inbox application to consume tasks from another TCM-compliant OData Provider.</td>
</tr>
<tr>
<td>For more information, see Configure My Inbox to Consume Tasks from Another TCM-Compliant OData Provider [page 21].</td>
</tr>
</tbody>
</table>

**Table 20:**

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Properties pane in the Workflow editor is vertically aligned.</td>
</tr>
</tbody>
</table>
Table 21:

New

In the Monitor Workflows app as well as in the REST API, you can also search workflow instances by their business keys.

For more information, see Manage Workflows Using the Monitor Workflows App [page 22].

Table 22:

New

Workflow errors are displayed in the Problems view of SAP Web IDE Full-Stack.

For more information, see Define Workflows [page 38].

Table 23:

New

You can view and edit scripts used in workflow using the Code Editor of SAP Web IDE Full-Stack.

For more information, see Configure Script Tasks [page 51].

Table 24:

Enhanced

Minor updates and bugfixes.

Table 25:

New

With the REST API, you can change the context of a workflow instance to recover erroneous instances.

For more information, see

- Modifying the Context of a Workflow Instance [page 89]
- Using Workflow APIs [page 85]
15 August 2017 (CW33) - Workflow Service

Table 26:

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can model an intermediate message event using workflow editor. Intermediate message events are process steps where the respective workflow instance waits for a message before the flow commences in the respective control flow branch.</td>
</tr>
</tbody>
</table>

For more information, see:
- Configure Intermediate Message Events [page 40]
- Define Workflows [page 38]
- Accelerated Modeling with Speed Buttons [page 61]

Table 27:

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>With the REST API, you can send message events to specific workflow instances.</td>
</tr>
</tbody>
</table>

For more information, see
- Authorization Configuration [page 99]
- Workflow Execution Log [page 93]

Table 28:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can use not only concrete user assignments for user tasks, but also group assignments, containing multiple users.</td>
</tr>
</tbody>
</table>

For more information, see Configure User Tasks [page 43].

8 August 2017 (CW32) - Workflow Service

Table 29:

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can claim and release tasks in My Inbox.</td>
</tr>
</tbody>
</table>

Table 30:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>With the REST API, you can suspend and resume a workflow instance.</td>
</tr>
</tbody>
</table>

For more information, see Using Workflow APIs [page 85].
### Important Note

“Claim” actions on task instances and “Suspend” actions on workflow instances reflect in the REST APIs. For example, in instance lists or execution logs. If you have built applications that use these APIs, you should check whether the new statuses are handled appropriately.

---

**25 July 2017 (CW30) - Workflow Service**

### Table 32:

**Enhanced**

Workflow admins have permissions to retrieve the workflow logs for a given workflow instance.

---

**18 July 2017 (CW29) - Workflow Service**

### Table 33:

**New**

The business key is a nontechnical unique identifier of a workflow instance. Workflow developers can model the business key in the workflow editor, which is then displayed in the Monitor Workflows app to administrators.

For more information, see Define Workflows [page 38] (SAP Web IDE workflow editor) and the standalone workflow editor.

### Table 34:

**New**

The workflow service is available as a free tile for your trial account.

For more information, see Signing Up for a Trial Account.

---

**13 July 2017 (CW28) - Workflow Service**

### Table 35:

**New**

You can model workflows using the workflow editor feature in SAP Web IDE.

For more information, see Modeling a Workflow [page 32].
### 11 July 2017 (CW28) - Workflow Service

**Table 36:**

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor updates and bugfixes.</td>
</tr>
<tr>
<td>For more information, see Workflow Service API documentation.</td>
</tr>
</tbody>
</table>

### 4 July 2017 (CW27) - Workflow Service

**Table 37:**

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>With the REST API or the Monitor Workflows app, workflow administrators have the following options:</td>
</tr>
<tr>
<td>● Page workflow instances</td>
</tr>
<tr>
<td>● Use free-text backend search</td>
</tr>
<tr>
<td>For more information, see Using Workflow APIs [page 85].</td>
</tr>
</tbody>
</table>

**Table 38:**

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>With the REST API workflow administrators can filter workflow instances.</td>
</tr>
<tr>
<td>For more information, see Using Workflow APIs [page 85].</td>
</tr>
</tbody>
</table>

**Table 39:**

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the execution log of a workflow instance you can inspect the recipient users of a user task as well as the error message of a failed script or service task. These details are available in the Monitor Workflows app and when using the REST API.</td>
</tr>
<tr>
<td>For more information, see Using Workflow APIs [page 85].</td>
</tr>
</tbody>
</table>

**Table 40:**

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each user task you can decide whether to display the respective execution log information in the inbox.</td>
</tr>
</tbody>
</table>
27 June 2017 (CW26) - Workflow Service

Table 41:

### Enhanced

The task count on the My Inbox tile is refreshed automatically every 15 seconds. The administrator has to republish the content in the SAP Fiori launchpad to make this change available.

20 June 2017 (CW25) - Workflow Service

Table 42:

### New

You can inspect details of a service task in the execution history of a workflow instance. These details are available in the Monitor Workflows app and when using the REST API.

For more information, see Workflow Service API documentation.

Table 43:

### Enhanced

You can see the number of tasks for processing by you in the My Inbox tile.

13 June 2017 (CW24) - Workflow Service

Table 44:

### New

You can inspect the execution history of a workflow instance by viewing the execution log displayed as part of the Monitor Workflows app.

For more information, see Manage Workflows Using the Monitor Workflows App [page 22].

7 June 2017 (CW23) - Workflow Service

Table 45:

### Enhanced

Minor runtime bug fixes
30 May 2017 (CW22) - Workflow Service

Table 46:

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can inspect the execution history of a workflow instance by viewing the workflow log in My Inbox. Moreover, you can use the REST API to integrate the execution history into your custom application.</td>
</tr>
<tr>
<td>The execution history starts with this release. Workflows started before this date will have an incomplete history. Some history entries will be missing, for example, the \textit{Workflow Started} entry. An incomplete history has no impact on the execution of the workflow.</td>
</tr>
<tr>
<td>For more information, see \textit{Using Workflow APIs} [page 85] and \textit{Workflow Execution Log} [page 93].</td>
</tr>
</tbody>
</table>

23 May 2017 (CW21) - Workflow Service

Table 47:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor documentation corrections</td>
</tr>
</tbody>
</table>

16 May 2017 (CW20) - Workflow Service

Table 48:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor runtime bugfixes</td>
</tr>
</tbody>
</table>

9 May 2017 (CW19) - Workflow Service

Table 49:

<table>
<thead>
<tr>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can navigate to a section of the workflow using the diagram overview in the workflow service editor.</td>
</tr>
</tbody>
</table>

Table 50:

<table>
<thead>
<tr>
<th>Enhanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>The speed buttons available while modeling a workflow are grouped together under tasks and gateways.</td>
</tr>
<tr>
<td>For more information, see \textit{Accelerated Modeling with Speed Buttons} [page 61].</td>
</tr>
</tbody>
</table>
Table 51:

New
You can view the technical IDs of the workflow artifacts, which helps you uniquely identify the artifact during deployment failure.

25 April 2017 (CW17) - Workflow Service

Table 52:

New
In destinations, service tasks support the proxy type OnPremise. Developers can use version 2 of SAP Cloud Platform cloud connector to route the service calls to on-premise systems.
For more information, see Destinations [page 101].

Table 53:

New
Administrators also have the permissions to retrieve the context of a workflow instance using the REST API.
For more information, see Using Workflow APIs [page 85].

4 April 2017 (CW14) - Workflow Service

Table 54:

Enhanced
You can use context variables using JUEL expressions in a unified way in the workflow service editor.
For more information, see Expressions [page 62].
2 Feature Scope Description

2.1 About This Document

This document defines the functional scope of SAP Cloud Platform Workflow.

Further restrictions may apply based on your license agreement with SAP. Functions and capabilities described in the documentation of this product may exceed the functional scope of the product to explain the integration with other SAP products, which must be licensed separately.

Target Groups

- Developers: Create workflows.
- Administrators: Assign users to roles and monitor workflows. Troubleshoot errors in the workflow.
- Workflow participants: Process tasks in the inbox.

2.2 Feature Scope Description

The SAP Cloud Platform Workflow offers state-of-the-art process automation capabilities.

This lightweight workflow service orchestrates automated steps and steps requiring user interaction. Typical examples for user interaction are single or multi-step approvals or requests for form-based user input.

Features

The workflow service offers an array of workflow capabilities.

Table 55: Key Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model workflows</td>
<td>Using the workflow editor in SAP Web IDE Full-Stack based on the Business Process Model and Notation editor you design workflows with a graphical user interface.</td>
</tr>
<tr>
<td>Manage workflow instances</td>
<td>Either with an API or with a user interface, you can manage workflows and tasks. You can, for example, start, cancel, or retry workflows.</td>
</tr>
</tbody>
</table>
### Feature Scope Description

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process tasks in an inbox</td>
<td>Workflow participants can view their assigned tasks in their inboxes and complete them.</td>
</tr>
</tbody>
</table>

### 2.3 Browser Support

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

**Note**
The workflow editor does not support Microsoft Internet Explorer and Safari.

<table>
<thead>
<tr>
<th>Table 56: Supported Browsers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
</tr>
<tr>
<td>Microsoft Internet Explorer</td>
</tr>
<tr>
<td>Mozilla Firefox</td>
</tr>
<tr>
<td>Google Chrome</td>
</tr>
<tr>
<td>Safari</td>
</tr>
</tbody>
</table>

### 2.4 Availability

The workflow service is available in the following languages.

- Workflow editor: English
- Workflow application: English
- 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
3 Administration

Introduction [page 18]
Configure the Workflow Service [page 18]
Manage Workflows Using the Monitor Workflows App [page 22]
Export Workflow Service Data [page 26]

3.1 Introduction

Configuration tasks for the SAP Cloud Platform Workflow service.

Related Information

Configure the Workflow Service [page 18]
Manage Workflows Using the Monitor Workflows App [page 22]

3.2 Configure the Workflow Service

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

Prerequisites

- A global account in your respective region. For more information, see Getting a Global Account.
- Your user is a member of the subaccount and is assigned to the Administrator role for the subaccount. You need this role to execute the following configuration steps. For more information, see Subaccount Member Roles.
Procedure

1. In the SAP Cloud Platform cockpit, enable the SAP Cloud Platform Portal and SAP Cloud Platform Workflow services for your subaccount.
   a. In the navigation area, choose Services.
   b. Search for SAP Cloud Platform Portal.
   c. On the Portal tile, choose Enable.
2. Search and enable the SAP Cloud Platform Workflow service.
   This automatically performs the following:
   ○ Enables principal propagation. Please do not disable it. For more information, see Principal Propagation [page 101] Principal Propagation in Destinations [page 101].
   ○ Creates the two destinations ‘bpmworkflowruntime’ and ‘iwsworkspaceruntime’. Please do not change them.

   **Note**
   Both configurations are required to run the workflow service.

3. Decide which roles or permissions your users need, then assign those roles and permissions.
   For more information about the available roles and permissions, see Authorization Configuration [page 99].
   Assign your users to workflow roles.
   1. On the Workflow tile, choose Configure Service.
   2. In the navigation area, choose Roles.
   3. In the Roles table, select the role that you want to assign to one or more users.
   4. You have the following options:
      ○ To assign the role to an individual user, choose Assign in the Individual Users table.
      ○ To assign the role to a group of users, choose Assign in the Groups table, and enter the name of a group.
   For more information, see Configure SAP Fiori Launchpad Objects [page 20].

Related Information

SAP Cloud Platform Portal documentation
3.2.1 Configure SAP Fiori Launchpad Objects

As an administrator, you can import SAP Fiori launchpad objects shared by the workflow service. These objects include the Workflow and My Inbox catalogs.

Prerequisites

- The TENANT_ADMIN role assigned to your user. To check which roles are assigned to you, see Check the Roles Assigned to You [page 21]. For more information, see Services in the Cockpit in the SAP Cloud Platform documentation.
- An SAP Fiori launchpad site on your subaccount. For more information about creating sites, see the SAP Fiori Launchpad Sites documentation. For more information about creating content, see Creating Content in the Launchpad Configuration Cockpit in the SAP Cloud Platform, portal service documentation.

Context

The configuration used for this procedure is only a sample. Depending on your requirements, the catalog assignments and groups created might look different.

Procedure

1. Choose or create a site and prepare it to use standard workflow service content.
   a. In the navigation area of the SAP Cloud Platform cockpit, choose Services > Portal Service.
   b. On the Portal Service tile, choose Enable, and then Go to Service.
   c. In the navigation area, choose Site Directory.
   d. Hover over the existing SAP Fiori launchpad site, and choose Edit.

2. To distribute the apps to all users, assign a role, for example, Everyone to the existing workflow catalog.
   You can restrict access to apps using groups, so that, for example, only administrators can access the Monitor Workflow and all users can access My Inbox.
   a. In the navigation area, choose Content Management.
   b. In the navigation area, choose Catalogs.
   c. Select your Workflow Catalog, and choose Edit.
   d. Choose Roles, and then choose the plus icon.
   e. Select a role from the dropdown list, for example, Everyone.
   f. Confirm with OK, and choose Save.

3. For your users to see the newly created content, publish your site by choosing (Publish Site) in the upper right corner.
4. To open the newly configured site, choose Site Directory, hover over the SAP Fiori launchpad site, and then select the link on the site tile.

   This is the link you typically share with your users so they can access the apps.

Results

The selected or created group name along with the app appears on the home page of the site.

3.2.1.1 Check the Roles Assigned to You

To configure the SAP Fiori launchpad objects you need the TENANT_ADMIN role.

Procedure

1. In the navigation area of the SAP Cloud Platform cockpit, choose Services Portal Service.
3. In the navigation area, choose Roles.
4. Select the TENANT_ADMIN role, and verify that your user is listed under User ID.
5. If your user ID is not listed, then assign this role to your user.

3.2.1.2 Configure My Inbox to Consume Tasks from Another TCM-Compliant OData Provider

Context

By default, My Inbox application, as part of SAP Cloud Platform Workflow, is preconfigured to consume tasks from the SAP Cloud Platform Workflow Service.

You can configure My Inbox to connect to another TCM-compliant OData service, different from the SAP Cloud Platform Workflow Service.

This can be achieved by adding a new My Inbox application in the SAP Fiori Configuration Cockpit and maintaining the respective tcmURL parameter.
You can expose this application as a SAP Fiori launchpad tile, which launches a separate instance of My Inbox application connected to the configured TCM-compliant OData service.

### 3.3 Manage 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.
Prerequisites

- The SAP Fiori launchpad objects are configured. For more information, see Configure SAP Fiori Launchpad Objects [page 20].

Managing Workflow Instances

The workflow instances view shows a list of all running workflow instances.

The following actions are available:

- To search the workflow instances, use the following criteria: workflow ID, workflow definition ID, subject, business key, or the initiator of the workflow instance.
- To display details about a workflow instance and to navigate to it, select a workflow instance.
To retry the execution of failed steps of an erroneous workflow instance, choose *Retry* on the details screen of the workflow instances.

To cancel a running workflow instance, choose *Terminate* on the details screen of the workflow instances.

To load more entries, scroll down to the end of the list and choose *More*.

To suspend a running or erroneous workflow instance, choose *Suspend* on the details screen of the workflow instance.

To resume a suspended workflow instance, choose *Resume* on the details screen of the workflow instance. This also retries failed steps.

To navigate to the workflow definition of an instance, click the workflow definition ID on the details screen of the workflow instance.

### Managing 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 start a new workflow instance, select a workflow definition and choose *Start New Instance*. When you start a new workflow instance, you can pass context data in form of a JSON string.
Example

Start New Instance

Enter the JSON context with which to start the new instance:

```json
{
    "product": "Hamlet (Paperback)",
    "inStock": true,
    "inventory": 20000,
    "price": 7.49,
    "publishingDate": "1600-04-23T16:25:43.511Z",
    "author": { "name": "William Shakespeare" },
}
```

Start New Instance  Cancel

The JSON structure contains the content to be passed to the workflow context. In contrast to the workflow service API a context node as a wrapper is not required.

**Note**

In the workflow context, use numbers where computations or comparisons on them are required. We do not recommend to use numbers as IDs, especially not for business keys. Use a string instead.

For more information about using these actions, see [Workflow Service API documentation](#).

- To navigate to the list of all instances of a definition, select the definition from the list and choose **Show Instances**.
- 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.
3.4 Export Workflow Service Data

The export provides access to your business data stored within the workflow. You can use this data to address, for example, audit needs.

Prerequisites

You have the WorkflowTenantOperator role that allows you to export runtime data related to workflow definitions and workflow instances.

Context

Caution

The export does not contain technical details that are required to reimport the data to the workflow service.

There are two types of data that you can export from the workflow service:

- Design time or modeling artifacts from the workflow editor.
  For more information, see Transport Workflows between Accounts [page 60].
- Runtime artifacts from the workflow service using the workflow service API.
  You export data related to a workflow definition or a workflow instance. The exported data and format is based on the Workflow Service REST APIs.

The following procedure describes the export of the runtime artifacts.

Procedure

To export the data, enter the following URL: https://<host>/workflow-service/rest/v1/export.

For more information, see Determine the Service Host [page 88].

Results

Caution

To verify that the export completed successfully, please check that you can extract the zip archive. The archive should not contain a file named error-log.txt. If there is 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 `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.
- A `workflow-instance-data` folder: For each workflow instance on the system one file (`<workflow-instance-ID>.json`) is written. It contains the latest version of the context and the execution logs related to this instance.
4 Develop

Introduction [page 28]
General [page 28]
Modeling a Workflow [page 32]
Developing a User Interface for a Workflow [page 65]
Using Workflow APIs [page 85]

4.1 Introduction

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

Related Information

General [page 28]
Modeling a Workflow [page 32]
Developing a User Interface for a Workflow [page 65]
Using Workflow APIs [page 85]

4.2 General

The SAP Cloud Platform Workflow offers state-of-the-art process automation capabilities.

This lightweight workflow service orchestrates service calls and user interaction steps. Typical examples for user interaction are single or multi-step approvals or requests for form-based user input.

The workflow service is tightly integrated with the SAP Cloud Platform infrastructure. This is true for the concepts such as multitenancy and subscriptions, destinations, role assignments, and scaling and for the services, for example, SAP Fiori launchpad, My Inbox, and the SAP Cloud Platform Integration service.

For on-premise systems, customers can integrate all relevant parts of the service, including identity provider integration and system orchestration, using standard SAP Cloud Platform means, for example, the cloud connector.
Caution

SAP Cloud Platform Workflow does not provide any support for storing and processing personal or sensitive data in the workflow context. It is the responsibility of the developers on customer side to decide what kind of data is stored in the workflow context.

Features

The workflow service offers an array of workflow capabilities, for example:

- **Modeling a Workflow** [page 32]
  
  Use a Web-based Business Process Model and Notation (BPMN) editor to design workflows with the graphical user interface (UI).

- **Using Workflow APIs** [page 85]
  
  Manage workflow instances and workflow definitions either using the REST API or the Monitor Workflows app.

- **Working with Tasks in My Inbox** [page 95]
  
  Use the tasks you have created to process your workflow.
The Workflow Service in a Simplified Landscape

The end user and the developer at the customer site work on subscriptions of the workflow service and SAP Web IDE Full-Stack. The workflow service itself resides in the SAP Cloud Platform subaccount.

1. The developer at the customer site creates an application, which can include multiple services, in the SAP Cloud Platform customer subaccount.
2. In the SAP Cloud Platform customer subaccount, the developer accesses the SAP Web IDE Full-Stack and enables the workflow feature to create workflows.
3. The developer accesses his or her browser to define a start event in the editor and start the workflow using the REST API or the Monitor Workflows app.
4. The end users at the customer site can access the workflow tasks in their My Inbox apps in the SAP Fiori launchpad.
5. In general, the customer application can call the workflow service APIs, for example, to start a new instance. At the same time, the workflow service can call the services of the application that is defined in the customer subaccount.

Figure 1: How the Workflow Service is Embedded into the Landscape
Related Information

Business Process Model and Notation

4.2.1 Workflow Definition versus Workflow Instance

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

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

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

The actual execution of these actions is called a workflow instance. So, a single workflow definition can have multiple workflow instances. This differentiation is essential for monitoring and troubleshooting. Additionally, you can define a subject for a workflow that helps the business users to track these instances using monitoring application. For more information, see Manage Workflows Using the Monitor Workflows App [page 22].

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

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

4.2.2 Conventions

Variable Names in the Process Context

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"
• Have to start with a letter (latin alphabet, or A-Z, a-z)
• Can contain additional letters, digits, and underscores

Duration

When expressions are used to specify duration, they must resolve to ISO 8601 format during runtime. For more information, see ISO 8601.

However, the following points must be considered:

• The smallest units which the duration specification supports are minutes.
• The “Week” unit (“W”) is not supported.
• The duration specification supports integers only.

4.3 Modeling a Workflow

You can model a workflow using the workflow editor in SAP Web IDE Full-Stack, which enables IT specialists to create workflows using the graphical Business Process Model and Notation (BPMN) standard. The BPMN workflow model also allows users to describe the flow of activities, events, and decision gateways.

Note

• The workflow editor is available only in regions where the SAP Cloud Platform Workflow service is offered.
• The workflow editor does not support Microsoft Internet Explorer and Safari.

Modeling a workflow includes the following steps, which you can perform using workflow editor 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 40].
• 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 and service or script tasks that are performed by the system. For more information, see Tasks [page 42].
  ○ Gateways: Gateways control the flow of execution in a workflow. For more information, see Gateways [page 57].
• Defining an endpoint of the process: Defines an endpoint of the process using end event or terminate end event. For more information, see Events [page 40].

Related Information

Enable the Workflow Editor in SAP Web IDE [page 33]
Importing Workflow Models [page 33]
Create a New Workflow Project [page 35]
Deploy Workflows [page 60]
Accelerated Modeling with Speed Buttons [page 61]
Expressions [page 62]
4.3.1 Enable the Workflow Editor in SAP Web IDE

Prerequisites

You have enabled the SAP Web IDE Full-Stack version.

Procedure

1. Log in to the SAP Web IDE application. You must use the SAP Web IDE Full-Stack version. For more information, see Opening SAP Web IDE.
2. From the left sidebar, open the Preferences perspective by choosing (Preferences).
3. Choose Features.
4. Enable the Workflow Editor feature by using the toggle.
5. Choose Save.
6. Reload SAP Web IDE by choosing Refresh.

4.3.2 Importing Workflow Models

You have a subscription to workflow service, and are modeling workflows using the standalone workflow service editor. You must perform the following steps to import workflows from standalone workflow service editor to the workflow editor in SAP Web IDE Full-Stack version:

- Rename JSON file to workflow file
- Configure the destination

Related Information

Rename JSON File to Workflow File [page 34]
Configure the Destination [page 34]
4.3.2.1 Rename JSON File to Workflow File

Import a JSON workflow model from the workflow service editor and convert it to a .workflow file, which makes the model consumable in SAP Web IDE.

Prerequisites

Create a workflow project. For more information, see Create a New Workflow Project [page 35].

Procedure

1. Log in to SAP Web IDE.
2. In the context menu of the project, choose File ➤ Import ➤ From File System.

   i Note

   The name of the JSON file you want to convert cannot contain any spaces. For example, you’ll be unable to convert Employee Onboarding Scenario.json; you must rename it without using spaces, for example, EmployeeOnboardingScenario.json.

3. In the Import dialog, browse for the .JSON file from local file system.
4. Choose the folder where you want to import the .JSON file.
5. Choose OK.
   
   The .JSON file appears under the folder.
6. In the context menu of the .JSON file, choose Rename.
7. In the Rename File dialog, change the name of the file to extension .workflow.
8. In the context menu of the renamed workflow, choose Open With ➤ Workflow Editor.

4.3.2.2 Configure the Destination

If you have subscribed to the workflow service, and are using the standalone workflow service editor, you must manually add the additional properties while configuring destinations in the SAP Cloud Platform Workflow cockpit to complete the importing process.

Context

To deploy workflows from the SAP Web IDE, add the following information.
Procedure

1. Log in to your subaccount in the SAP Cloud Platform cockpit.
2. In the navigation area, choose \(\text{Connectivity \rightarrow Destinations}\).
3. Choose the \(\text{bpmworkflowruntime}\) destination.
4. In the \(\text{Destination Configuration}\) section, choose \(\text{Edit}\).
5. In \(\text{Additional Properties}\), select \(\text{New Property}\), then choose \(\text{WebIDEEnabled}\) as the new property.
6. Set this property to \(\text{true}\).
7. Choose \(\text{Save}\).

4.3.3 Create a New Workflow Project

Context

A workflow project can hold one or more workflows. We recommend that you package all workflows for one scenario into a single project. You can only deploy workflows created within this project; that is, you cannot deploy the workflow project itself.

Procedure

1. Log in to the SAP Web IDE application.
2. From the left pane, choose \(\text{(Development)}\) and navigate to the \(\text{Workspace}\) folder.
3. Choose \(\text{File \rightarrow New \rightarrow Project from Template}\).
4. From the \(\text{Template Selection}\) screen, choose \(\text{Category}\) as \(\text{Business Process Management}\).
5. Choose the \(\text{Workflow Project}\) tile, then choose \(\text{Next}\).
6. On the \(\text{Basic Information}\) screen, enter a project name, then choose \(\text{Next}\).
7. In the \(\text{Workflow Details}\) screen, provide a workflow name and optional description.
8. Choose \(\text{Finish}\).

Note

To create multiple workflows, you can select a workflow project or the workflow folder and choose \(\text{New \rightarrow Workflow}\). By providing the name for the workflow, you can create another workflow within a project or the workflow folder.

Recommendation

We recommend that you create workflows in the \(\text{workflow}\) folder.
Results

The project wizard creates a project structure in the workspace. The project contains a workflow folder with a new sample workflow file. The workflow file contains the name that you have provided in the previous steps.

4.3.3.1 Open Workflow Files in the Workflow Editor

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

Procedure

1. Log in to the SAP Web IDE application.
2. From the left pane, choose (Development), and navigate to the Workspace folder.
3. Right-click the workflow file and choose Open With Workflow Editor.
4.3.3.2 Editor Layout

The workflow editor consists of the following areas:

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

You use this procedure to define workflows.

Procedure

1. Open the workflow with the Workflow Editor.
   For more information, see Open Workflow Files in the Workflow Editor [page 36].
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 employee onboarding process, you can consider a Subject like "Employee onboarding process initiated for ${context.employeename}". For more information, see Expressions [page 62].

   Note
   ○ The data that is referenced from the Subject using expressions, should be provided as payload during the start of the workflow instance. For more information, see Workflow Definition versus Workflow Instance [page 31].
   ○ 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 85].

3. In the Business Key field of the Workflow Properties pane, provide an optional identifier for workflow instances based on business data.
   The business key can include static text as well as expressions similar to the workflow subject. With the business key, you can later identify a workflow instance without knowing the technical instance ID.

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

   Note
   In SAP Cloud Platform Workflow uniqueness is not 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. To model the start event of a workflow, select Events > Start Event and drop it onto the canvas from the palette.
5. In the **Start Event Properties** pane, provide a name and documentation for the start event.

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

6. To add a task to the workflow, see **Tasks** [page 42].
7. To add a gateway to the workflow, see **Gateways** [page 57].
8. To add an intermediate message event, see **Configure Intermediate Message Events** [page 40].
9. To add an intermediate timer event, see **Configure Intermediate Timer Events** [page 41].

10. To connect two flow elements, choose the icon from the first flow element to be connected first, keep the mouse button pressed and move your cursor to the next flow element that needs to be performed in the workflow.

   **Note**
   If you choose a flow element using the speed button, and the connection automatically appears. In this case the above step is not required.

   For more information about speed buttons, see **Accelerated Modeling with Speed Buttons** [page 61].

11. To model the end event of a workflow, choose ![Events](https://example.com) ![End Event](https://example.com) and drop it onto the canvas from the palette.
12. In the **End Event Properties** pane, provide a name and documentation for the end event.
13. To model the end of a workflow as a terminate end event, choose ![Events](https://example.com) ![Terminate End Event](https://example.com) and drop it onto the canvas from the palette.

    For more information on the terminate end event, see **Events** [page 40].

    **Note**
    You can also model a terminate end event using the speed buttons. For more information, see **Accelerated Modeling with Speed Buttons** [page 61].

14. In the **Terminate End Event Properties** pane, provide a name and description for terminate end event.
15. To format the workflow model, choose ![Arrange Horizontally](https://example.com) ![Arrange Vertically](https://example.com) from the toolbar.
16. Choose **Save**.

   **Recommendation**
   We recommend that you save the changes before exiting. If you do not, your changes will be lost.

   **Note**
   - To rename any of the flow objects, select the flow object and choose ![F2](https://example.com).
   - You can view the errors while designing a workflow in the **Problems** view. For more information, see **Using the Problems View**.
   - Choose ![Undo](https://example.com) or ![Redo](https://example.com) from the toolbar to undo or redo an action. Alternatively, you can choose the following key combinations:
4.3.4.1 Events

An event affects the flow of the process.

SAP Cloud Platform Workflow editor supports the following events:

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

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

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

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

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

4.3.4.1.1 Configure Intermediate Message Events

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

**Prerequisites**

Configure a business key for your workflow. For more information about business keys, see Define Workflows [page 38].

**Context**

Clients can send messages via the REST endpoint. For more information about how to send messages, refer to Workflow Service API Reference.

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

1. Select Events Intermediate Message and drop it onto the canvas from the palette.
2. In the Intermediate Message Event Properties area, choose the General tab.
3. Fill in the Name and Documentation fields for the intermediate message event.
4. In the Intermediate Message Event Properties area, choose the Details tab.
5. In the Message Name field, provide a name of the message.
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).
- If you don’t provide a response variable, the message is consumed by matching workflow instances. However, the context data passed by the message is not considered.

7. Choose Save.

**Example**

Equipment must be procured for a new hire. In this case, the employeeID of the new hire can be configured as business key. The workflow calls an external service to trigger the asynchronous procurement process. The workflow instance must wait until the procurement process is completed.

You can model an intermediate message event, which blocks the execution of the workflow in this branch until a message is received. When the procurement process completes, the external system can send a message that includes details about the equipment ordered. This message is then delivered to one of the waiting workflow instances, and the execution moves to the next flow step.

**4.3.4.1.2 Configure Intermediate Timer Events**

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

**Context**

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

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

   **Note**
   Provide an expression in the **Duration** field using ISO 8601 format. For example, `PT${(context.minutes)}M`. The JUEL expression `${(context.minutes)}` is evaluated at runtime. You can provide multiple duration attributes by using multiple JUEL expressions. For more information about the duration formats that are supported in ISO 8601, see **Conventions** [page 31].
   - To use a static value, choose **Static Value** from the **Duration Based On** dropdown. Now, provide the **Duration** as a numeric value, and choose a **Unit of Time**.

   **Note**
   If you are using a static value, make sure you don’t select the **Use Expression** option.
5. Choose **Save**.

**4.3.4.2 Tasks**

SAP Cloud Platform Workflow editor supports the following tasks:

- **User Task**: A user task is a flow object in the process that illustrates a task that a human performs. These tasks then appear in My Inbox where processor of the task can complete the task instance, and view description of the task.
- **Service Task**: A service task is a flow object in the process that illustrates a task that a system performs, for example, calling an external service. The task that you configure is performed immediately, when the process execution arrives at the service task.
- **Script Task**: A script task is a flow object in the process that illustrates a script that gets executed when the process execution arrives at the script task. This is an automated activity.

**Related Information**

- Configure User Tasks [page 43]
- Configure Service Tasks [page 48]
- Configure Script Tasks [page 51]
4.3.4.2.1 Configure User Tasks

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

Prerequisites

The following configuration steps are required to ensure that end users can view tasks in custom UIs in My Inbox:

- Deploy a custom task UI application and ensure that it is up and running in the consumer subaccount.
- Ensure that the application contains the SAPUI5 component, which is used as custom task UI.

Context

As a workflow developer, you must be able to associate a custom task UI in the customer application with a workflow user task. In this way, when an end user opens his or her task on My Inbox, the custom task UI is rendered.

Procedure

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

   - A unique ID gets generated for every workflow artifact. This ID is in read-only mode.
   - Ensure the Name field is short, precise and contains sufficiently unique identifier, as it is displayed to the end users. For example, in My Inbox.

5. (Optional) To display information about the task execution in the inbox workflow log, select Show in inbox workflow log.
6. From User Task Properties area, choose the Details tab.
7. Depending on the priority of the user task, choose one of the following options from the Priority menu:
   - Low
   - Medium (default)
   - High
   - Very High
8. In the **Display Texts** section, provide the following details
   - **Subject**: Title of the task instance.
   - **Description**: Any additional information.

9. In the **Recipients** section, provide the unique IDs of the users or name of groups of users in **Users or Groups** who should process the task.

10. Configure a custom task user interface. For more information, see [Configure a Custom Task User Interface](page 46).

11. To configure the duration by when the task is due, select the **Configure Due Date** checkbox.
    
    You must configure the due date by following the below substeps:
    1. To provide a duration for the due date as an expression, choose **Expression** from the **Due Date Based On** dropdown. Now, provide the due date in the **Duration** field as an expression.
    
    **Note**
    You must provide an expression in the **Duration** field using a subset of the ISO 8601 format. For example, `PT${context.minutes}M`. The JUEL expression `${context.minutes}` is evaluated at runtime. You can provide multiple duration attributes by using multiple JUEL expressions. For more information about the duration formats that are supported in ISO 8601, see [Conventions](page 31).
    2. 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**.

    **Note**
    Due date is reflected in My Inbox using which the end user can sort and filter the tasks. For more information, see [Working with Tasks in My Inbox](page 95).

12. (Optional) Add a boundary timer event to include a timer for the user task. For more information, see [Configure Boundary Timer Events](page 45).

13. Connect the user task to the required flow elements.

14. Choose **Save**.
4.3.4.2.1.1 Configure Boundary Timer Events

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

Context

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

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

Example

In an employee onboarding scenario, the equipment assignment to a new hire must be confirmed by the buddy assigned to the new hire. The buddy is responsible for confirming the equipments that need to be procured for the new hire.

A non-canceling boundary timer event can be modeled on the "Confirm or Change Equipment" user task to send a reminder mail to the buddy if the task is not completed in three days. Similarly, a canceling boundary timer event can be modeled where the duration is such that the timer elapses two days before the joining date of new hire. Additionally, an alternative escalation flow, such as an escalation email, must be sent to the manager of the buddy to take required action; in this case, the original "Confirm or Change Equipment" task becomes irrelevant. Hence, the 'Confirm or Change Equipment' user task is canceled.

Procedure

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

**Note**

You must provide an expression in the Duration field using a subset of the ISO 8601. For example, `PT${context.minutes}M`. The JUEL expression `${context.minutes}` is evaluated at runtime. 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 [page 31].

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

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

**Note**

○ You can add multiple boundary timer events to a user task, which gets triggered when the corresponding timers are fired. When a canceling boundary event is triggered, any boundary events attached to the same task that haven’t yet triggered are canceled.

○ One specific case needs to be taken into account: namely, suspending and resuming a workflow instance with several boundary timer events on an active user task. If such an instance is resumed and it has been suspended for a time period longer than the corresponding timer durations, there is no deterministic order in which the events are triggered.

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

### 4.3.4.2.1.2 Configure a Custom Task User Interface

**Procedure**

1. To embed a custom task UI for projects that are available in the workspace, perform the following:
   a. In the User Interface section, choose the Select option.
   b. 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 Cloud Platform.

c. Choose SAPUI5 Component path from the dropdown menu.
d. Choose OK.

**Note**

SAPUI5 Component is added automatically, which is editable.
2. To manually provide the custom task UI details, provide following details in the *User Interface* section:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML5 App</td>
<td>Name of the HTML5 application</td>
</tr>
<tr>
<td>Component URL</td>
<td>Location of <code>&lt;Component.js&gt;</code> in the HTML5 project</td>
</tr>
<tr>
<td>SAPUI5 Component</td>
<td>SAPUI5 component name without the <code>.component</code> suffix</td>
</tr>
</tbody>
</table>

There are two examples provided based on whether the Grunt build is triggered for the SAPUI5 project or not. The configuration of the above *User Interface* properties is illustrated below with examples.

**Example**

Example 1: Grunt build is not enabled for the SAPUI5 project.

Open the component.js file of the UI5 application. The sample screenshot is given and the corresponding *User Interface* properties are shown in the table below:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML5 App</td>
<td><code>employeeonboarding</code></td>
</tr>
<tr>
<td>Component URL</td>
<td><code>webapp</code></td>
</tr>
<tr>
<td>SAPUI5 Component</td>
<td><code>sap.demo.Waas</code></td>
</tr>
</tbody>
</table>

**Example**

Example 2: An SAPUI5 project is built using Grunt. For more information, see Building Applications. The sample screenshot for this and the corresponding *User Interface* properties are shown in the table below:
3. Save the workflow.

Related Information

Configure User Tasks [page 43]

4.3.4.2.2 Configure Service Tasks

Configure a service task if you want the system to perform a particular task in the workflow.

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 Documentation for service task. 
   
   Documentation is optional.

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

5. In the Service Task Properties area, choose the Details tab.
6. Provide the following details:
   - Destination: Name of a destination in the consumer subaccount, which determines the host to connect to at runtime. For more information on the supported feature set of destinations, see Destinations [page 101].
   - Path: Resource path that appends to the URL of the specified destination while calling the service.

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

   Note
   - If the HTTP method is POST, DELETE, or PUT, then the Path to XSRF Token field appears. XSRF token is used for modifying operations that are protected against XSRF (cross-site request forgery) attacks. For more information, see SAP Cloud Platform.
   - Path to XSRF Token: The resource path that needs to be appended to a specified destination, while calling the service to fetch an XSRF token.
   - 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.

   Note
   - The referenced node is used 1:1 as content for the request body.
   - If the HTTP Method is POST and PUT, then the Path to XSRF Token field appears.
   - Response Variable: Link to a workflow context node that is created or overwritten to finally store the body of the HTTP response.
Note
The referenced node is used to store the response body.

Example

This example shows how to call a REST service to store employee's leave requests. This service is XSRF protected.

The service url for this example is https://{host}/leaverequest

- **Destination** is created in the SAP Cloud Platform 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

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

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

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

7. Connect the service task to the required flow elements.
8. Choose \textit{Save}.

\textbf{Next Steps}

The workflow developer can deploy the workflow model into the workflow service runtime. To make the workflow operational, the administrator must create and configure the destination mentioned by the workflow developer. For more information, see \textit{Destinations [page 101]}.

\textbf{Related Information}

\textit{Accelerated Modeling with Speed Buttons [page 61]}

\textbf{4.3.4.2.3 Configure Script Tasks}

Configure a script task if you want the system to run a script to perform a task in the workflow.

\textbf{Context}

\begin{itemize}
  \item \textit{Note}
    If you have previously modeled script task using the workflow editor, then your existing script files are converted into .js files automatically. Create a JavaScript file only for new script tasks you want to model.
  \item \textit{Recommendation}
    We recommend that you export and import workflow projects, rather than individual workflows, as additional script resources are added to the workflow project.
\end{itemize}

\textbf{Procedure}

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

\begin{itemize}
  \item \textit{Note}
    A unique read-only \textit{ID} is automatically generated for every workflow artifact.
\end{itemize}
3. Perform one of the following steps to add JavaScript files:
   - Choose Select to browse for the JavaScript file in the current project.
   - To create a JavaScript file, perform the following steps:
     1. Choose the Create JavaScript File link.
     2. In the New File window, provide a filename.

   **Note**
   The filename must have a ".js" extension.

3. Choose OK.
4. In the JavaScript file, provide the script.

**Note**
- You can view/edit the JavaScript file by selecting the Script File link.
- For more information about Code Editor, see Developing Applications.
- The provided APIs, as well as the objects and arrays stored in the workflow context, are non-native JavaScript objects; that is, ECMAScript host objects. Their behavior might differ from that of the native objects. For more information about supported APIs, see:
  - Creating and Reading Workflow Context Structures [page 52]
  - Accessing Contextual Information During Execution of Script Tasks [page 56]
- The script must be in JavaScript that is based on ECMAScript 5.1. For more information, see the Ecma Web page. Restrictions: 'eval' and 'Function' are not supported for script tasks. Using the function keyword is supported, but you cannot assign functions to workflow context variables.
- The execution of script tasks is subject to resource limits, for example, with respect to processing time or memory usage. If the limits are exceeded, the corresponding workflow instance is put into erroneous state. The error is written to the error logs of the workflow instance. You can retrieve the error logs either using the REST API or the Monitor Workflows app.

4. Choose OK.
5. Save the workflow.

**Related Information**

Accelerated Modeling with Speed Buttons [page 61]
Transport Workflows between Accounts [page 60]

**4.3.4.2.3.1 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

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

Reading Variables

```javascript
// 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
}
```

Setting Variables

```javascript
// 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

```javascript
// the following will remove the variable from the context
delete $.context.myString;
```
Manipulating the Context Directly

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

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

Related Information

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

4.3.4.2.3.1.1 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

```javascript
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"
}
```

### 4.3.4.2.3.1.2 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

```javascript
// 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

```javascript
// 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
} else 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

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

4.3.4.2.3.2 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.

Script tasks cannot modify the $.info API. All its properties are provided by SAP Cloud Platform Workflow and are read-only.

Getting Information About the Environment

```javascript
var workflowInstanceId = $.info.workflowInstanceId; // for example, "42"
var taskDefinitionId = $.info.taskDefinitionId; // for example, "ScriptTask1"
var workflowDefinitionId = $.info.workflowDefinitionId; // for example, "scriptTaskProcess"
```

4.3.4.2.4 Configure a Mail Task

Context

Procedure
4.3.4.3 Gateways

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

SAP Cloud Platform Workflow 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 are defined. The sequence flow with a condition that evaluates to true is selected for continuing the process. If multiple sequence flows have a condition that evaluates to true, the first one defined is selected for continuing the process. If none of the conditions defined for the sequence flow evaluate to true, then the one marked as default flow is selected and the execution proceeds along that path.

  **Note**

  If you use an exclusive gateway to split flow into multiple sequence flows, then the same type of gateway should be used to merge as well.

  For more information, see Configure an Exclusive Gateway [page 58].

- **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 is 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**

  Parallel gateway works on a logical level, it does not speed up the technical execution.

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

  **Note**

  If you use a parallel gateway to split flow into multiple paths, then the same type of gateway should be used to merge as well.

  For more information, see Configure a Parallel Gateway [page 59].
4.3.4.3.1 Configure an Exclusive Gateway

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

Procedure

1. From the palette, choose <Gateways>Exclusive Gateway</Gateways> drop it on to the canvas.
2. In the <Exclusive Gateway Properties> area, provide a <Name> and <Documentation> for the gateway.
   
   **Note**
   
   A unique ID gets generated for every workflow artifact. This ID is in read-only mode.

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

4. Configure a condition for a sequence flow.
   
   For more information, see Configure a Sequence Flow [page 58].

5. Choose <Save>.

Related Information

Accelerated Modeling with Speed Buttons [page 61]

4.3.4.3.1.1 Configure a Sequence Flow

Procedure

1. On the canvas, choose the sequence flow you want to configure.
2. In the <Sequence Flow Properties> section provide a <Name> and <Documentation> for the flow object.
   
   **Note**
   
   A unique ID gets generated for every workflow artifact. This ID is in read-only mode.
3. From an exclusive gateway, provide a **Condition** to the outgoing sequence flow or mark the sequence flow as **Default**.

   **Note**
   - You can mark only one outgoing sequence flow as the default.
   - If you want a certain path to execute, for example, only if an employee does not belong to Germany. You need to configure the sequence flow condition as `$({context.employee.region} != "Germany")`. For more information, see Expressions [page 62].

4. Choose **Save**.

### 4.3.4.3.2 Configure a Parallel Gateway

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

**Procedure**

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

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

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

**Related Information**

[Accelerated Modeling with Speed Buttons](page 61)
4.3.5 Transport Workflows between Accounts

Transport workflows from one subaccount to another account.

Procedure

1. In your workspace, choose Export from the context menu of the workflow project.
   A local .zip file is created for your project.
2. Import the workflow projects to the target account.
   For more information, see Importing Projects from an Archive.
3. Deploy all the workflows in the workflow project. For more information, see Deploy Workflows [page 60].

   Note
   You must also use this procedure to import any custom task UIs that are used in workflows.

4.3.6 Deploy Workflows

You use this procedure to deploy modeled workflows.

Prerequisites

You have the WorkflowDeveloper role. For more information, see Authorization Configuration [page 99].

Procedure

1. Ensure that you have completed modeling the workflow.
   A new workflow instance automatically uses the latest deployed version of a given workflow definition.
   Workflow instances that have been started with a previous version of the workflow definition are not affected by this change.
2. In the context menu of the workflow file in the project explorer, choose Deploy to SAP Cloud Platform Workflow

   Note
   • Ensure that you save the workflow before deploying a workflow.
4.3.7 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:

<table>
<thead>
<tr>
<th>Speed Button</th>
<th>Description</th>
</tr>
</thead>
</table>
| ![End Event](image) | This speed button allows you to model the following events:  
  - ![End Event](image): Creates an end event.  
  - ![Intermediate Message](image): Creates an intermediate message event.  
  - ![Intermediate Timer](image): Creates an intermediate timer event. |
| ![User Task](image) | This speed button allows you to model the following tasks:  
  - ![User Task](image): Creates a user task.  
  - ![Service Task](image): Creates a service task.  
  - ![Script Task](image): Creates a script task. |
### 4.3.8 Expressions

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

Expressions are mainly used for the following purposes:

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

The expressions you use must adhere to the syntax defined by the Java Unified Expression Language (JUEL). The syntax to access data in the workflow context within a JUEL expression and via the Script Task API is aligned. The following statements would address the same attribute stored in the workflow context:

- `context.employee.firstname` (JUEL expression)
- `.context.employee.firstname` (script task API)

For more information about the script task API, see Creating and Reading Workflow Context Structures [page 52].
Property Navigation and Text Expression

Property navigation and text expressions typically occur in user tasks. See Configure User Tasks [page 43].

Example

Sample Code

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

Examples that are supported by expression syntax include the following:

- **Property navigation (including text expressions without static texts)**
  - Accessing the `name` property of the `employee` context variable (dot notation): `context.employee.name`
  - Accessing the `name` property of the first entry in the `peers` array property of the `employee` context variable: `context.employee.peers[0].name`

- **Text expressions**
  - Combining static with dynamic content: Dear `context.employee.name`

- **Conditions**
  - Applying Boolean operators to form a condition: `context.employee.region!= "Germany" && context.employee.isManager == true`

Conditions and Variable Specifications

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

- **Condition expressions** have to evaluate to a Boolean value, that is true or false.
- **You must specify conditions like the ones used in the example in Configure a Sequence Flow [page 58]** 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 48])** 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 48]) as follows:
Notices

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

Overview of Properties Supporting JUEL Expressions

Table 58: Elements and Properties Using JUEL Expressions

<table>
<thead>
<tr>
<th>Workflow Model Element Type</th>
<th>JUEL-Enabled Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence flow originating from an exclusive gateway</td>
<td>Condition</td>
</tr>
<tr>
<td>Service task</td>
<td>Path</td>
</tr>
<tr>
<td>Service task</td>
<td>Path to XSRF token</td>
</tr>
<tr>
<td>User task</td>
<td>Subject</td>
</tr>
<tr>
<td>User task</td>
<td>Recipient users</td>
</tr>
<tr>
<td>User task</td>
<td>HTML5 App</td>
</tr>
<tr>
<td>User task</td>
<td>Component URL</td>
</tr>
<tr>
<td>User task</td>
<td>SAPUI5 Component</td>
</tr>
<tr>
<td>Workflow model</td>
<td>Subject</td>
</tr>
</tbody>
</table>

Related Information

JUEL Tutorial - Expression Language
4.4 Developing a User Interface for a Workflow

As the workflow service includes REST-based APIs that let you access the workflow service runtime, you can develop scenario-specific user interfaces (UIs) on top of these APIs.

The main use cases for such UIs include the following:

- **Start UI**: Triggers new workflow instance for a defined workflow definition.
- **Task UI**: Is plugged into My Inbox to represent a user task in the workflow definition.

Both types of UIs can be developed and deployed as HTML5 applications on SAP Cloud Platform. For more information about developing HTML5 applications using the workflow editor, see SAP Web IDE.

The following diagram depicts the relationships between the involved HTML5 applications and the respective subscriptions for My Inbox and the workflow service runtime. The different applications and subscriptions are wired using destinations.

My Inbox includes two predefined routes, which you can use when developing UIs:

- **/bpmworkflowruntime**
  - /bpmworkflowruntime maps to the bpmworkflowruntime destination, which is configured by default for your subaccount.
  - For more information, see Read Task Context Data [page 68].
  - For more information about routes, see Application Descriptor File.

- **/html5apps**
  - You can integrate the UIs into any HTML5 app and access them using /html5apps/<name_of_app>.

![Figure 3: Custom UI Overview](image)

**Related Information**

- Custom Task UI [page 66]
- Creating Start UI [page 76]
4.4.1 Custom Task UI

With the custom task user interface (UI), endusers can access their workflow tasks in their inboxes.

Build a custom task UI by doing the following:

1. Creating an HTML5 Application for the Custom Task UI [page 66]
2. Deploy an HTML5 Application [page 75]

Related Information

SAP Web IDE Full-Stack Documentation
HTML5: Getting Started
Task Data in the SAP Cloud Platform, Workflow Service API Documentation

4.4.1.1 Creating an HTML5 Application for the Custom Task UI

This is an overview of the series of steps you have to execute to create the custom task UI.

Procedure

1. Create an HTML5 application. For more information, see Creating an HTML5 Application in the SAP Cloud Platform documentation.
2. Create a project using the SAPUI5 Application template.
   For more information, see Creating a Project in the SAP Cloud Platform documentation but use XML as view type.
3. Extend the application by modifying the webapp/Component.js file by doing the following:
   a. Get the Task Instance ID [page 68]
   b. Read Task Context Data [page 68]
   c. Bind a UI Element to an Attribute of the Task Context JSON Model [page 69]
   d. Adding Task Completion Buttons [page 70]
   e. Get the Task Description [page 72]
Results

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

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

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

```
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("/bpmworkflowruntime/rest/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:

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

SAP Cloud Platform Workflow
Develop
PUBLIC 67
4.4.1.1.1 Get the Task Instance ID

Procedure

To get the task ID, add the following lines to the init function:

Sample Code

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

4.4.1.1.2 Read Task Context Data

Procedure

1. Read the task context data via a REST service and create a JSON model from it.

   For more information, see Task Data in the SAP Cloud Platform, Workflow Service API Documentation.

2. After the model has been created, set it as default model of the component so it can be used for data binding.
### 4.4.1.1.3 Write Task Instance Data

**Procedure**

1. Write the task context data via a REST service and create a JSON model from it.
   
   For more information, see Task Data in the SAP Cloud Platform, Workflow Service API Documentation.

2. After the model has been created, set it as default model of the component so it can be used for data binding.

### 4.4.1.1.4 Bind a UI Element to an Attribute of the Task Context JSON Model

**Procedure**

1. To display a field of the task context on the custom task UI, add a text element to `webapp/view/<view name>.view.xml` and bind it to the text attribute of the JSON model.

2. Replace the page element with this content:

   ```xml
   <Page showHeader="false" showFooter="false">
       <content>
           <Text text="{/text}" maxLines="0" id="__text0"/>
       </content>
   </Page>
   ```
4.4.1.1.5 Adding Task Completion Buttons

This is an overview of the series of steps you have to execute to add task completion buttons.

To add these buttons, execute the following steps:

1. Fetch an XSRF Token [page 70]
2. Call the Task Completion REST API [page 71]
3. Add Approve and Reject Buttons [page 71]
4. Update the Task List After Task Completion [page 72]

4.4.1.1.5.1 Fetch an XSRF Token

Procedure

To call the task completion REST API, you have to retrieve an XSRF token first. You could, for example, use the following function:

Sample Code

```javascript
_fetchToken: function() {
  var token;
  $.ajax({
    url: "/bpmworkflowruntime/rest/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;
}
```
4.4.1.1.5.2 Call the Task Completion REST API

Procedure

1. Call the previously created `_fetchToken` function.
2. Using this token, call the completion API with data, which will be written into the task or workflow context.

**Example**

In this example, the data contains a field named `approved` to indicate whether the task was approved or rejected.

**Sample Code**

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

4.4.1.1.5.3 Add Approve and Reject Buttons

**Procedure**

To add buttons to the footer of the custom task UI, add these lines to the init function of `webapp/Component.js`:

**Sample Code**

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

The previously created function `_completeTask` is called in both actions but with different approval status.

### 4.4.1.1.5.4 Update the Task List After Task Completion

**Procedure**

1. Refresh the Master view of the inbox after a task has been completed, so the completed task disappears from the list.
   
   Add, for example, the following lines:
   
   ```javascript
   _refreshTask: function(taskId) {
     this.getComponentData().startupParameters.inboxAPI.updateTask("NA", taskId);
   }
   ```

2. Call this function after the task has been completed.

### 4.4.1.1.6 Get the Task Description

To retrieve the task description:

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

```xml
<ObjectAttribute title="Description" text="{/context/task/Description}"/>
```

The `init` function of `webapp/Component.js` should be like this:
4.4.1.1.7 Show and Hide Footer

To hide footer within the page:

The onInit function of the view controller should be like this:

```javascript
onInit: function() {
  var startupParameters = this.getComponentData().startupParameters;
  startupParameters.inboxAPI.setShowFooter(false);
},
```

4.4.1.1.8 Showing and Hiding the Back Navigation Button

You can show, hide, and customize the Back navigation button in the header of a custom task UI.

- To show the Back navigation button in the header of a custom task UI, your onInit function of the View controller should be the following:

```javascript
onInit: function() {
  var startupParameters = this.getComponentData().startupParameters;
  startupParameters.inboxAPI.setShowFooter(false);
},
```
Passing only the first parameter `true` will start the default button handler, which will execute `window.history.back()`.

In some cases, this does not lead to the desired behavior. Therefore, you need to provide a custom handler, as shown in the next example:

- If you want to further customize your back navigation handler, you can edit your `onInit` function of the `View` controller as follows:

```javascript
onInit: function() {
  var startupParameters = this.getComponentData().startupParameters;
  startupParameters.inboxAPI.setShowNavButton(true, function(){
    alert("You are about to leave this task");
    window.history.back();
  });
},
```

- To hide the Back navigation button in the header of a custom task UI, your `onInit` function of the `View` controller should be the following:

```javascript
onInit: function() {
  var startupParameters = this.getComponentData().startupParameters;
  startupParameters.inboxAPI.setShowNavButton(false);
},
```

### 4.4.1.1.9 Call an External Service from a Custom Task UI

To show, for example, contextual information that is not available in the workflow context, you want to call the REST service, which you developed yourself, from within a custom task UI.

#### Prerequisites

Your REST service is deployed on SAP Cloud Platform or is reachable from your customer account.

#### Procedure

1. In your HTML5 application containing the custom task UI, define an additional route in the `neo-app.json` file, which targets a destination pointing to your service, for example, deployed as a Java application on SAP Cloud Platform.
2. In your account that is subscribed to the workflow service, create a new destination with the name you specified in the previous step.

3. Configure the destination against your deployed service.

   **Note**

   If you want to propagate the user from My Inbox to your REST service, select App2App SSO as the authentication type to use.

4. In your custom task UI application, call your REST service using an Ajax call. The service is then available at the following URL: /html5apps/<taskui_application>/<destination_name>/<relative_api_path>

   **Sample Code**

   ```javascript
   _callService: function() {
     $.ajax({
       url: "/html5apps/custtakui/external-service/v1/external-data",
       method: "GET",
       contentType: "application/json",
       async: false,
       data: ""
     });
   }
   ```

### 4.4.1.2 Deploy an HTML5 Application

You deploy your HTML5 app using standard SAP Cloud Platform procedures.

**Procedure**

To activate your application deploy it to SAP Cloud Platform.

For more information, see **Deploying Your App to SAP Cloud Platform** in the SAP Cloud Platform documentation.
4.4.2 Creating Start UI

Create a sample UI for starting workflow instances.

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

Prerequisites

A workflow definition of interest is deployed into the workflow service runtime.

Procedure

1. Create an HTML5 Application for the Start UI [page 76]
2. Define the Destination Route [page 77]
3. Extend the View [page 77]
4. Extend the Controller [page 78]

Related Information

SAP Web IDE Full-Stack Documentation
HTML5: Getting Started

4.4.2.1 Create an HTML5 Application for the Start UI

You create your HTML5 app using standard SAP Cloud Platform procedures.

Procedure

1. Create an HTML5 application.
   For more information, see Creating an HTML5 Application in the SAP Cloud Platform documentation.
2. Create a project using the SAPUI5 Application template.
   For more information, see Creating a Project in the SAP Cloud Platform documentation.
4.4.2.2 Define the Destination Route

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

Procedure

In the neo-app.json file created in the webapp folder of your application, include the following destination route element pointing to workflow service runtime into the routes array:

Sample Code

```json
{
  "path": "bpmworkflowruntime",
  "target": {
    "type": "destination",
    "name": "bpmworkflowruntime",
    "entryPath": "/workflow-service"
  },
  "description": "Workflow Service Runtime"
}
```

4.4.2.3 Extend the View

Context

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

Procedure

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

Sample Code

```xml
<Page title="Workflow Start UI">
  <content>
    <VBox width="100%" direction="Column" id="__vbox0">
      <items>
        <Input width="100%" id="textInput" value="{/text}"/>
      </items>
    </VBox>
  </content>
</Page>
```
4.4.2.4  Extend the Controller

Procedure

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

- Implement Data Model Instantiation [page 79]
- Implement XSRF Token Fetch [page 80]
- Implement Workflow Instance Start [page 80]
- Bind an Action to the Button Push Event [page 81]

Results

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

```
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");
    $.ajax({
      url: "/bpmworkflowruntime/rest/v1/workflow-instances",
      method: "POST",
      async: false,
      contentType: "application/json",
      headers: {
        "X-CSRF-Token": token
      }
    });
  },
}
```
4.4.2.4.1 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: ''
  }));
}
```

4.4.2.4.2 Implement XSRF Token Fetch

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

```javascript
_sampleCode
_fetchToken: function() {
  var token;
  $.ajax({
    url: "/bpmworkflowruntime/rest/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;
}
```

4.4.2.4.3 Implement Workflow Instance Start

The workflow is started using the corresponding HTTP call to the workflow service REST API, see Workflow Service API documentation. In this example, the input of the user is used in the context of the workflow instance: namely, in its text field. In addition, the response of the call is assigned to the corresponding property in the data model:

```javascript
_samplesCode
_startInstance: function(token) {
  var model = this.getView().getModel();
  var inputValue = model.getProperty("/text");
  $.ajax({
    url: "/bpmworkflowruntime/rest/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));
    }
  });
}
```
4.4.2.4.4  Bind an Action to the Button Push Event

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

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

### 4.4.3  Data Propagation from My Inbox to Task Application

You can associate a custom task UI to a workflow user task, which is then rendered in the detail view of My Inbox for the task.

When you select a task, My Inbox instantiates the custom task UI application component and transfers a set of data for the selected task.

#### Sample Code

```javascript
startupParameters{
    taskModel: JsonModel > {{InstanceID:<value>, TaskDefinitionID:<value>,......}},
    applicationPath: <string>,
    queryParameters: {<key>:<value>,<key>:<value>,...... },
    inboxAPI:{<APIs>}
}
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskModel</td>
<td>sap.ui.model.json.JSONModel</td>
<td>Contains task properties. For more information, see Accessing Task Data from Task Application [page 82].</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>queryParameters</td>
<td>Json</td>
<td>Contains key value pairs of query parameter and value configured in the Task UI component integration configuration.</td>
</tr>
<tr>
<td>applicationPath</td>
<td>single-value string data</td>
<td>Contains the path to the Task UI component using which the component is loaded.</td>
</tr>
<tr>
<td>inboxAPI</td>
<td></td>
<td>Contains the API published by My Inbox for integration purposes.</td>
</tr>
</tbody>
</table>

For more information, see My Inbox UI Integration API Reference [page 83].

---

**From the Task UI component, this data can be accessed as shown below:**

```javascript
var startupParameters = this.getOwnerComponent().getComponentData().startupParameters;
```

---

### 4.4.3.1 Accessing Task Data from Task Application

Task data is transferred from My Inbox to the task application on startup using the property `taskModel`. `taskModel` is of type `sap.ui.model.json.JSONModel` and contains the following task properties:

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

**PriorityText** and **StatusText** contain translated texts that are specific to the My Inbox user’s locale.
4.4.3.2 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.

Table 60: Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>object</td>
<td>A JSON object specifying action details. Properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● action: string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● label: string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● type: string (either Accept or Reject)</td>
</tr>
<tr>
<td>actionEventHandler</td>
<td>function</td>
<td>The function to be called when the event occurs.</td>
</tr>
<tr>
<td>listener?</td>
<td>object</td>
<td>Context object to call the event handler.</td>
</tr>
</tbody>
</table>

**Return Value**

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

**getDescription**

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

Table 61: Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAPOrigin</td>
<td>string</td>
<td>Value for the parameter SAP__Origin for the specific task</td>
</tr>
<tr>
<td>taskInstanceId</td>
<td>string</td>
<td>Value for the parameter InstanceId for the specific task</td>
</tr>
</tbody>
</table>

**Return Value**

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

Removes an action added previously by the integrated application.

Table 62: Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionName</td>
<td>string</td>
<td>Name of the action to be removed</td>
</tr>
</tbody>
</table>

**Return Value**

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

**updateTask**

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

Table 63: Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAPOrigin</td>
<td>string</td>
<td>Value for the parameter SAP__Origin for the specific task</td>
</tr>
<tr>
<td>taskInstanceId</td>
<td>string</td>
<td>Value for the parameter InstanceId for the specific task</td>
</tr>
</tbody>
</table>

**Return Value**

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

**setShowFooter**

Shows or hides footer of the page.

Table 64: Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showFooter</td>
<td>boolean</td>
<td>Flag representing whether to show or hide footer in the page. The default value is false.</td>
</tr>
</tbody>
</table>

**setShowNavButton**

Shows or hides navigation button in header of the page.
### 4.5 Using Workflow APIs

The REST-based API allows a tight integration of tasks on SAP Cloud Platform with SAP Cloud Platform Workflow.

With the workflow service you, as a developer, can integrate workflow-related operations into your applications. The workflow service API supports the following functions:

- 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.

For more information about who can execute these actions, see Authorization Configuration [page 99].

For more information about using these actions, see the Workflow Service API documentation on the product page or the SAP Cloud Platform Workflow API hub documentation.

Clients must authenticate to use workflow service APIs. The following authentication types are supported:

- Basic authentication.
- SAML2
- OAuth2 (client credentials, authorization code, and SAML 2.0 BearerAssertion Flow for OAuth 2.0)

#### Related Information

- Manage Workflows Using the Monitor Workflows App [page 22]
- Access Workflow APIs Using OAuth 2.0 Authentication (Client Credentials) [page 86]
- Access Workflow APIs Using OAuth 2.0 Authentication (Authorization Grant) [page 87]
4.5.1 Access Workflow APIs Using OAuth 2.0 Authentication (Client Credentials)

Call workflow service APIs using OAuth 2.0 authentication (client credentials flow).

Prerequisites

- Create a new client in SAP Cloud Platform cockpit for your subaccount using the following data:

  Table 66:
<p>|</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription</td>
<td>&lt;SAP provider account&gt;/bpmworkflowruntime</td>
<td>Creates the OAuth 2.0 client in the context of your workflow service subscription.</td>
</tr>
<tr>
<td>Authorization Grant</td>
<td>Client Credentials</td>
<td>Specifies the OAuth 2.0 flow that is used to request the access token and authenticate the API call.</td>
</tr>
</tbody>
</table>

For more information, see Register an OAuth Client in OAuth 2.0 Configuration.

- Assign the necessary role of the workflow service API that you want to use to call to a user named oauth_client_<clientID>. For more information about roles, see Authorization Configuration [page 99].

Context

i Note

The workflow service does not define any OAuth 2.0 scopes. Instead, assign the existing roles to the user who executes the service calls.

Procedure

1. 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 client credentials. To determine the endpoint URL in the cockpit, see Security » OAuth » Branding » OAuth URLs.
      Example: https://oauthasservices-<your_account>.<landscape_host>/oauth2/api/v1/token?grant_type=client_credentials
   b. Authenticate the call using basic authentication, where the user name corresponds to your OAuth client ID and the password to the client secret.
c. Copy the access token from the HTTP response.

2. Perform the call to the workflow service API by sending the access token as the header:
   ○ Header name: Authorization
   ○ Header value: Bearer <access token>

4.5.2 Access Workflow APIs Using OAuth 2.0 Authentication (Authorization Grant)

Call workflow service APIs using OAuth 2.0 authentication (authorization grant flow).

Prerequisites

- Create a new client in SAP Cloud Platform cockpit for your subaccount using the following data:

  Table 67:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription</td>
<td>&lt;SAP provider account&gt;/bpmworkflowruntime</td>
<td>Creates the OAuth 2.0 client in the context of your workflow service subscription.</td>
</tr>
<tr>
<td>Authorization Grant</td>
<td>Authorization Code</td>
<td>Specifies the OAuth 2.0 flow that is used to request the access token and authenticate the API call.</td>
</tr>
</tbody>
</table>

For more information, see Register an OAuth Client in OAuth 2.0 Configuration.

- Assign the necessary role of the workflow service API that you want to call to the user on whose behalf the call to the workflow service API is executed. Typically, this is the user who authenticates the call to the OAuth 2.0 authorization endpoint below. For more information about roles, see Authorization Configuration [page 99].

Context

Procedure

1. Request an access token from the OAuth 2.0 authorization server.
   a. Send a GET request to the token endpoint and specify both the client ID and the response type as "code".
      To determine the endpoint URL in the cockpit, see Security > OAuth > Branding > OAuth URLs.
Example: https://oauthasservices-<consumer_account>.<landscape_host>/oauth2/api/v1/authorize?client_id=<clientId>&response_type=code

b. Authenticate the call using the real user that should be propagated to the workflow service API (on behalf user).

The access token is sent to the URL that you specified as callback URL in the client details.

c. Copy the access token from the HTTP response.

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 client credentials. To determine the endpoint URL in the cockpit, see Security > OAuth > Branding > OAuth URLs.

Example: https://oauthasservices-<your_account>.<landscape_host>/oauth2/api/v1/token?grant_type=authorization_code&code=<access_token>

b. Authenticate the call using basic authentication, where the user name corresponds to your OAuth client ID and the password to the client secret.

c. Copy the access token from the HTTP response.

3. Perform the call to the workflow service API by sending the access token as the header:

   ○ Header name: Authorization
   ○ Header value: Bearer <access token>

4.5.3 Determine the Service Host

The URL of the host has the following format: https://<host>/workflow-service/rest. To work with the API actions, you must determine the specific URL.

Context

Note

If you access the workflow APIs from an HTML5 app, you need to use another URL. Instead of directly referring to the subscription URL defined in the destination shown above, you have to refer to an HTML5 destination route.

If you develop a custom task UI, you can leverage the existing destination route in My Inbox. For an example of how to access the workflow API from a custom task UI, see Creating an HTML5 Application for the Custom Task UI [page 66].

If you develop an HTML5 app from scratch, you have to define a new destination route on your own. For more information, see Define the Destination Route [page 77].
Procedure

1. In the cockpit, choose your subaccount.
2. Choose Connectivity Destinations in the navigation area.
3. From the list of destinations, select bpmworkflowruntime.
4. Under Destination Configuration, find the URL link.
5. Copy the URL, and use it in your API URL. Make sure that the complete URL ends with /rest.

4.5.4 Modifying the Context of a Workflow Instance

You can modify a 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 ensure that further entries are not written into the corresponding execution log. Then the context modification is considered safe from collisions with any ongoing workflow instance activities. After the necessary changes to the context are performed, you can resume the workflow instance execution. See the section about suspending or resuming workflow instance. For more information, see /v1/workflow-instances/{workflowInstanceId}.

Override Context

Overriding a context of the workflow instance removes the contents of the context before performing the override operation. It is substituted with the payload of the operation.

Example

Context contents before overriding:

```json
{
    variableOnlyInOldContext: 1,
    variableOverriden: "good bye!",
    variableNestedObject: {
        variableNested: true,
        variableNestedInOldContext: 1000
    }
}
```

Override operation payload:

```json
{
    variableOverriden: "hello!",
    variableNestedObject: {
        variableNested: false,
        variableNestedNew: "new value"
    }
}
```
Patch Context

Patching a context of the workflow instance merges the contents of the context before performing the override operation with the payload of the operation.

The following situations are possible in this case:

- A variable is present in the workflow instance context and in the operation payload. After the operation is performed, the value of this variable in the workflow instance context is equal to the corresponding value in operation payload.
- A variable is present in the workflow instance context, but not in the operation payload. After the operation is performed, the variable remains unchanged.
- A variable is not present in the workflow instance context before performing the operation, but it is present in the operation payload. After the operation is performed, the variable is added in the workflow instance context with the corresponding value.

Note

Merging happens at all levels of complex objects nesting.

Example

Context contents before patching:

```
{
  variableOnlyInOldContext: 1,
  variableOverriden: "good bye!",
  variableNestedObject: {
    variableNested: false,
    variableNestedNew: "new value"
  },
  variableNew: "I'm new"
}
```
Patch operation payload:

```json
{
  variableOverriden: "hello!",
  variableNestedObject: {
    variableNested: false,
    variableNestedNew: "new value"
  },
  variableNew: "I'm new"
}
```

Context contents after the override operation:

```json
{
  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 [page 31].

### 4.5.5 Updating Task Properties

With the task patch API, you can modify the properties of the tasks in status READY or RESERVED. Currently, updating the task subject and the task description are supported.

To update a task, you need to send an HTTP request with the **PATCH** method to the corresponding API endpoint with the following payload:

#### Example

**Task Update Payload**

```json
{
  "subject": "<New subject>",
  "description": "<New description>"
}
```
where <New subject> and <New description> refer to the values of the task subject and description after the operation is performed.

Expressions

You can use Expressions [page 62] to refer to the context of the relevant workflow instance while updating the task properties:

Example

Task Update Payload with Expressions

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:

Example

Workflow Instance Context

Sample Code

```
{
    "employee": {
        "name": "John",
        "surname": "Dow"
    },
    "price": 8000,
    "saleReduction": 0.5
}
```

The task will have the subject Approve purchase order for John Dow and the description Price: 4000 EUR.

Simultaneous Updating and Completing Tasks

With the same API endpoint that is used for updating the tasks you can also complete the tasks (link). To this end, "status" ("COMPLETED") and optionally "context" need to be present in the payload, for example:

Sample Code

```
{
    "context": {
        "price": 6000,
        "reductionReason": "Outdated"
    }
```
If you want to update and complete the task with the same request, the payload looks as follows:

```json
{
    "context": {
        "price": 6000,
        "reductionReason": "Outdated"
    },
    "status": "COMPLETED",
    [HCPWORKFLOWSERVICE-2248 "subject": "Approve purchase order for ${context.employee.name} ${context.employee.surname}",
    [HCPWORKFLOWSERVICE-2250 "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 taking into account the new values of the context. And, finally, 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 Dow", but the description is set taking into account the new values: "Price: 3000 EUR".

### 4.5.6 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 is not a technical log.

<table>
<thead>
<tr>
<th>Log Entry/Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKFLOW_STARTED</td>
<td>Workflow instance started.</td>
</tr>
<tr>
<td>WORKFLOW_COMPLETED</td>
<td>Workflow instance completed.</td>
</tr>
<tr>
<td>WORKFLOW_CANCELED</td>
<td>Workflow instance canceled.</td>
</tr>
<tr>
<td>WORKFLOW_SUSPENDED</td>
<td>Workflow instance suspended.</td>
</tr>
<tr>
<td>WORKFLOW_CONTINUED</td>
<td>Workflow instance continued after processing was stopped due to an error.</td>
</tr>
<tr>
<td>WORKFLOW_RESUMED</td>
<td>Workflow instance resumed.</td>
</tr>
<tr>
<td>Log Entry/Event</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>WORKFLOW_CONTEXT_OVERWRITTEN_BY_ADMIN</td>
<td>Context administrator completely overrode the workflow context.</td>
</tr>
<tr>
<td>WORKFLOW_CONTEXT_PATCHED_BY_ADMIN</td>
<td>Context administrator partially modified the workflow context.</td>
</tr>
<tr>
<td>USERTASK_CREATED</td>
<td>User task created.</td>
</tr>
<tr>
<td>USERTASKCLAIMED</td>
<td>User task claimed.</td>
</tr>
<tr>
<td>USERTASK_COMPLETED</td>
<td>User task completed.</td>
</tr>
<tr>
<td>USERTASK_RELEASED</td>
<td>User task released.</td>
</tr>
<tr>
<td>USERTASK_CANCELED_BY_BOUNDARY_EVENT</td>
<td>User task canceled by a boundary timer event.</td>
</tr>
<tr>
<td>SERVICETASK_CREATED</td>
<td>Service task created.</td>
</tr>
<tr>
<td>SERVICETASK_COMPLETED</td>
<td>Service task completed.</td>
</tr>
<tr>
<td>SERVICETASK_FAILED</td>
<td>Service task failed.</td>
</tr>
<tr>
<td>SCRIPTTASK_CREATED</td>
<td>Script task created.</td>
</tr>
<tr>
<td>SCRIPTTASK_COMPLETED</td>
<td>Script task completed.</td>
</tr>
<tr>
<td>SCRIPTTASK_FAILED</td>
<td>Script task failed.</td>
</tr>
<tr>
<td>INTERMEDIATE_MESSAGE_EVENT_REACHED</td>
<td>Intermediate message event reached from a workflow instance.</td>
</tr>
<tr>
<td>INTERMEDIATE_MESSAGE_EVENT_TRIGGERED</td>
<td>Intermediate message event triggered for a workflow instance.</td>
</tr>
<tr>
<td>INTERMEDIATE_TIMER_EVENT_REACHED</td>
<td>Intermediate timer event reached in a workflow instance.</td>
</tr>
<tr>
<td>INTERMEDIATE_TIMER_EVENT_TRIGGERED</td>
<td>Intermediate timer event triggered for a workflow instance.</td>
</tr>
<tr>
<td>CANCELING_BOUNDARY_TIMER_EVENT_TRIGGERED</td>
<td>Boundary timer event triggered the cancellation of the attached user task and continued the alternative flow.</td>
</tr>
<tr>
<td>NONCANCELING_BOUNDARY_TIMER_EVENT_TRIGGERED</td>
<td>Boundary timer event triggered the alternative flow attached to it without canceling the attached user task.</td>
</tr>
</tbody>
</table>
5 User Guide

Introduction [page 95]
Working with Tasks in My Inbox [page 95]

5.1 Introduction

The user guide for the SAP Cloud Platform Workflow service is for end-users and key-users. End-users find information on:

- Working with Tasks in My Inbox [page 95]
- Accessing the My Inbox Application [page 96]

5.2 Working with Tasks in My Inbox

You can process tasks from workflow service within My Inbox. My Inbox application runs on the SAP Fiori launchpad. You can use My Inbox on your desktop or mobile.

A user task is a type of flow object in the process that appears in My Inbox. You can work on the task, complete the task instance and view description of the task.

My Inbox displays the following information about the workflow and tasks:

- Task title
- Tasks with status Ready and Reserved
- Tasks with priority

Key Features

- View the tasks that are assigned to you.
- Claim tasks.

i Note

When you claim a task, you become its processor and its other recipients will no longer see it in My Inbox. In this case, the status of the task changes from Ready to Reserved.
- View your current and available tasks.
- Display the count of tasks on the My Inbox tile in the SAP Fiori launchpad.
- Sort tasks by priority, due date, task title, and the user who created the task.
- Filter your tasks by priority, due date, status Ready, and creation date.

<i>Note</i>

Please note that the task list is limited to the first 1000 entries that match the filter.

- Group tasks by task title, priority, status, and by task type. The task type is the name of the user task in the workflow model defined in the editor.
- View task specific details.
- Release tasks where you are the processor.

<i>Note</i>

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. In this case, the status of the task changes from Reserved to Ready.

- Execute and complete tasks.

### 5.2.1 Accessing the My Inbox Application

**Prerequisites**

- An SAP ID user and access to an SAP Cloud Platform trial or global account. For more information, see [Getting a Global Account](#).
- Assign the relevant workflow service runtime roles to the SAP ID user. For more information, see [Authorization Configuration](#) [page 99]
- Subscribe to the SAP HANA Platform, Portal service.
- Enable My Inbox app in the SAP Fiori Launchpad for users to access the application. For more information, see [Configure SAP Fiori Launchpad Objects](#) [page 20].
6 Security Guide

This guide provides an overview of the security-relevant information that applies to the SAP Cloud Platform Workflow.

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

Related Information

Security Guide for SAP Cloud Platform

6.1 Architecture

The architecture of the workflow service comprises several components and subservices.
The workflow service includes the following subservices that are provisioned into the customer subaccount using the SAP Cloud Platform cross-subaccount subscription concept:

- Workflow editor in SAP Web IDE Full-Stack
- Workflow service runtime
- Monitor workflows
- My Inbox

For more information, see Multitenant Applications in the SAP Cloud Platform documentation.

Prerequisites for using the workflow service:

- A subscription to the Portal Service respectively the SAP Fiori launchpad is required to use My Inbox.
- While the workflow service runtime exposes a set of REST-based application programming interfaces (APIs) for managing workflow instances and task instances, the workflow editor, the Monitor Workflows app, and My Inbox provide user interfaces (UIs) only.
- Access to all subservices of the workflow service requires a valid user identity in the corresponding identity provider configured in the customer subaccount.
  For more information, see Identity Provider and Identity Management [page 98].
- All UIs offer single sign-on authentication based on SAML assertions. The APIs of the workflow service runtime can be accessed with SSO authentication using SAML or OAuth 2.0 as well as basic authentication. In addition, all APIs that can lead to data manipulation in the workflow service runtime are protected against cross-site request forgery (XSRF).
  For more information, see the API documentation of the REST-based API.

### 6.2 Identity Provider and Identity Management

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

All requests handled by the workflow service subscriptions are authenticated against the identity provider of the customer subaccount and authorized against the role assignments specified on the subscriptions in the customer subaccount. All users who need to interact with the various subservices of the workflow service must be available in the respective identity provider. You can replace the default SAP Cloud Platform Identity Authentication service with your own corporate identity provider.

**Note**

For authentication using SAML or OAuth 2.0, you can use an additional corporate identity provider. Requests that use basic authentication are still handled by the SAP Cloud Platform Identity Authentication service.

For more information about the concepts and the necessary configuration steps, see Identity and Access Management in the SAP Cloud Platform documentation.
6.3 Authorization Configuration

To assign the necessary roles to its users, the workflow service relies on the standard functionality of SAP Cloud Platform and the My Inbox application relies on the authorization management of the back-end system.

For information about assigning roles and permissions, see Configure the Workflow Service [page 18].

Roles for Accessing Workflow Service Runtime

Assign roles to specific users using the subscription to the workflow service runtime (Java application).

Table 69: Available Roles for Accessing the Workflow Service Runtime

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
</table>
| WorkflowDeveloper       | • Permission to use the workflow editor  
• Permission to query workflow definitions  
• Permissions to retrieve the current error messages of a workflow instance  
• Retrieve an XSRF (cross-site request forgery) token as a prerequisite to work on workflow definitions |
| WorkflowContextAdmin    | • Permissions to partially modify or completely override the workflow context  
• Retrieve an XSRF (cross-site request forgery) token as a prerequisite to work on workflow definitions |
| WorkflowInitiator       | • Permission to start workflow instances (using the API or the Monitor Workflows app)  
• Retrieve an XSRF (cross-site request forgery) token as a prerequisite to start workflow instances |
| WorkflowParticipant     | • View tasks in My Inbox, where the user is a recipient  
• Permissions to perform task operations:  
  ○ claim  
  ○ release  
  ○ call the task completion API  
• Retrieve an XSRF (cross-site request forgery) token as a prerequisite to work on tasks |
<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
</table>
| WorkflowAdmin       | • Permission to use the Monitor Workflows app  
|                     | • Permissions to query workflow definitions as well as query and cancel workflow instances  
|                     | • Permissions to retrieve the context of a workflow instance  
|                     | • Permissions to retrieve the tasks of a workflow instance  
|                     | • Permissions to retrieve the current error messages of a workflow instance  
|                     | • Permissions to retry the failed steps of an erroneous workflow instance  
|                     | • Permissions to suspend and resume a workflow instance for temporary suspension of processing  
|                     | • Permissions to retrieve the workflow logs for a given workflow instance  
|                     | • Permission to download the Workflow model in the Monitor Workflow app  
|                     | • Retrieve an XSRF (cross-site request forgery) token as a prerequisite to work on workflow instances                                                                                                     |
| WorkflowMessageSender | Permission to send a message to a set of workflow instances for consumption in intermediate message events                                                |
| WorkflowTenantOperator | Permission to export data                                                                                                                                  |

### Permission for Accessing Workflow Editor

Access to the workflow editor application is secured by permissions. The `WorkspaceUser` permission grants access to the workflow editor.

**Note**

You can create a group with the required users who can model and deploy workflows, and assign the following roles to the group:

- WorkflowDeveloper role
- A role with `WorkspaceUser` permission

For more information about assigning permissions, see [Configure the Workflow Service](page 18).

**Note**

Permissions are required to only access the standalone workflow service editor.

**Note**

If you configured your own custom identity provider ([Identity Provider and Identity Management](page 98)) you need to additionally change the role assigned for the workflow editor. For this, create a new role for the editor subscription in the SAP Cloud Platform cockpit and assign users from your custom identity provider to the new role.
1. To create the new role, choose the **Workflow Service** tile, then choose **Configure Workflow Editor** > **Roles** > **New Role**.

2. Assign this new role to the **WorkspaceUser** permission as described above.

For more information, see **Managing Roles and Permissions**.

**Related Information**

Accessing the My Inbox Application [page 96]

## 6.4 Destinations

The subservices communicate using predefined destinations in the customer subaccount. The destinations are generated and configured when enabling the workflow service in a customer subaccount.

### Principal Propagation

Communication between the different subservices uses principal propagation, which forwards the user who is logged on to the user interface to the workflow service runtime. This lets all requests that are sent to the workflow service runtime on behalf of the user (who initiated the request from the user interface) be posted.

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

For more information about the concepts and the necessary configuration steps, see **Application-to-Application SSO Authentication** in the SAP Cloud Platform documentation.

### Destination Configuration for Service Task

The workflow service supports outbound connectivity to orchestrate 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.

While the standard destination concept in SAP Cloud Platform can be used for this purpose, there are several limitations that apply to their usage in the context of the workflow service.

The following destination features are supported in SAP Cloud Platform Workflow:

- **Authentication type:** No Authentication and Basic Authentication
- **Server authentication (verification):** JDK default and custom trust stores
- **Supported proxy type:** Internet, OnPremise
Destination type:
○ Supports HTTP and HTTPS connectivity based on HTTP destinations in SAP Cloud Platform.
○ For an OnPremise destination you can optionally specify the LocationId property.

To connect to on-premise back-end systems, you can use the SAP Cloud Platform cloud connector. For more information on how to install and configure the SAP Cloud Platform cloud connector, see SAP Cloud Platform Connectivity in the SAP Cloud Platform documentation.

Use the standard SAP Cloud Platform mechanisms in the SAP Cloud Platform cockpit to configure destinations. For more information, see Configuring Destinations from the Cockpit.

**i Note**

For server verification additional properties that were configured at the destinations as described in Server Certificate Authentication are ignored. Consequently, you cannot turn off trust verification, and host names are always verified in strict mode.

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 maintained in the SAP Cloud Platform cloud connector as URL of the destination instead of the actual URL of the back-end system. The scheme of the specified URL must be http://, not https://.

While the configuration data of destinations are stored completely within the customer subaccount, the workflow service runtime must temporarily access this data when executing a workflow instance. This data is not persisted within the workflow service itself.

### 6.5 Data Protection and Data Privacy

The workflow service offers functions to help you fulfill the legally required data protection and privacy measures to data that is stored on behalf of customers. These measures include, for example, the "need to know" principle during support processes, or data isolation between tenants.

These measures apply to all technical subservices provided by SAP that make up the workflow service.

However, customers can define workflows that call external services that are outside SAP. These services must implement appropriate data protection and privacy measures to avoid externally exposing critical data from SAP systems. These services must also be integrated in an appropriate way into the workflows that are running on the workflow service.

**Caution**

SAP Cloud Platform Workflow does not provide any support for storing and processing personal or sensitive data in the workflow context. It is the responsibility of the developers on customer side to decide what kind of data is stored in the workflow context.

The workflow service generally supports customers who are executing workflows in which employees or other persons in a suitable contractual relation with the customer take part. Customers might require explicit consent from workflow participants outside of such contractual relations, if their data is to be processed through the
service. Depending on how the data that is generated while using the service is used, customers might need to require explicit consent from persons with contractual relations.

### Note

The workflow service is not a system of record; that is, it is not the authoritative source of records as is, for example, an Enterprise Resource Planning system.

Therefore, the following applies to the workflow service:

- Consider the workflow context to be a temporary storage. That is, data can remain in the workflow service only for shorter periods than defined by company policies or mandated by law in systems of record. Suitable data deletion mechanisms that conform with company policies and applicable laws must be supplied.
- Authorization checks applied to a data item when accessed using the workflow service may differ from checks performed when the item is accessed from a system of record, for example, with respect to role-based authorizations.

### Recommendation

Be aware of the following recommendations with respect to data protection and privacy:

- Configure destinations to use secure communication protocols, such as HTTPS, wherever possible.
- 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.
- SAP does not provide any support for storing and processing personal or sensitive data in the workflow context; the customer or developer is responsible for providing any of this support that is required.
7 Troubleshooting

When working with the workflow service, you may encounter issues that prevent access or affect performance.

Related Information

- End Users Can't Open SAP Fiori Launchpad Tiles [page 104]
- Error When Clicking "Go to Service" on Portal Tile [page 106]
- HTTP Status 403: User Doesn't Have Sufficient Privileges [page 107]
- Tasks Not Appearing [page 107]
- Error During Workflow Deployment in SAP Web IDE [page 107]
- No Permissions Granted [page 108]
- Error in JavaScript Files [page 108]

7.1 End Users Can't Open SAP Fiori Launchpad Tiles

Symptom

When SAP provides a new version of an HTML5 UI component, for example, the Monitor Workflows app or My Inbox, end users may no longer be able to open the tiles for these applications. Whether or not an updated tile can be opened depends on the user's HTML5 cache status in the SAP Fiori launchpad. The open action fails with an error message, for example: Could not open app. Please try again later.

Cause

The cache in the SAP Fiori launchpad service component references a previously used version of the affected HTML5 application. After an update, the previous version is no longer available.

Solution

Clear the cache using a user account that has permission to run the cockpit application and to edit the SAP Fiori launchpad configuration for the given tenant.
1. In your browser, open the SAP Cloud Platform cockpit for the affected account.
2. In the navigation area, choose Services, then choose the Portal Service tile.

3. On the Portal Service page, choose the Go to Service link.
4. In the navigation area, choose Site Directory. Select the affected site, and choose Edit.

5. In the navigation area, choose Settings. Choose Actions, then choose Clear HTML5 Application Cache.
6. Confirm the warning dialog by selecting Yes. 
You see the following confirmation message: 
“HTML5 applications updated to their latest versions successfully.” 

Result 
End users can now reload their SAP Fiori launchpad and use the tiles again.

7.2 Error When Clicking "Go to Service" on Portal Tile 

Symptom 
When you chose Go to Service on the Portal tile in the services overview of the SAP Cloud Platform cockpit, you see the following error: The site cannot be displayed due to insufficient privileges as shown below. 

Solution 
1. Navigate to your subaccount. For more information, see Navigate to a Subaccount. 
2. In the navigation area for your subaccount, choose Services. 
3. Search for Portal. 
5. In the navigation area, choose Roles. 
6. Select the TENANT_ADMIN role and verify that your user is assigned to this role. 
7. If your user is not assigned, choose Assign, enter your user ID, and choose Assign. 
8. To see the change, log off and log on again. 

Result 
When you now choose Go to Service on the Portal tile, the landing page of the Portal service opens.
7.3 HTTP Status 403: User Doesn't Have Sufficient Privileges

**Symptom**
You're missing appropriate permissions in the workflow service runtime.

**Solution**
- See the Authorization Configuration [page 99].
- Assign the Workflow Participant role to your user and all other users that are supposed to access My Inbox.

7.4 Tasks Not Appearing

**Symptom**
You can't see the tasks you've created, even though you directly assigned your user as a recipient user.

**Solution**
User IDs are case sensitive in the workflow service. When you authenticate against the SAP Cloud Platform Identity Authentication service, your user ID is provided in all uppercase letters. Your user ID must match the one in the recipient user field. Therefore, use only uppercase letters to enter your user ID, for example, P123456789 instead of p123456789.

7.5 Error During Workflow Deployment in SAP Web IDE

**Symptom**
You see the following error message while you are deploying the workflow editor on SAP Web IDE:

![Error message](image)

**Solution**
See the list of possible causes for the failure and the respective solutions:
- The workflow service isn't enabled. To enable the workflow service, see Configure the Workflow Service [page 18].
• The destination isn’t configured correctly. To configure the destination, see Configure the Destination [page 34].

Result
After performing these steps, you should be able to deploy the workflow again.

7.6 Error in JavaScript Files

Symptom
JavaScript files in the workflow project show an error on exceeding 10000 characters in the Problems view. Workflow project also contains JavaScript files that are not used in a script tasks of workflow.

Solution
A rule is applied to all the JavaScript files used in the workflow project. This rule throws an error when the JavaScript files exceed a character length of 10000. However, if you have JavaScript files that are not used in script tasks in the workflow, you can manually disable this validation rule by following these steps:

1. On the context menu of the workflow project, choose Project Settings.
2. From the Code Checking menu, choose JavaScript.
3. In the Rules section, disable the wfs_max-characters rule by using the toggle button.
4. Choose Save.

Result
The Problems view should no longer throw this error on JavaScript files.

7.7 No Permissions Granted

Symptom
You have assigned the appropriate roles in the SAP Cloud Platform cockpit; however, you still receive the following message: No permission (forbidden).

Solution
Either log off using an action usually available in the top right corner and log on again, or clear your browser cache to delete existing cookies.

New roles that are assigned to you are not applied to existing browser sessions; they do not take effect until after you log in again to the SAP Cloud Platform Identity Authentication service.
Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP’s gross negligence.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP’s gross negligence or willful misconduct. All links are categorized for transparency (see: https://help.sap.com/viewer/disclaimer).