SAP Cloud Platform Workflow

SAP Cloud Platform Workflow

Document Version: 1.0 – 2018-12-12
# Content

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
</tr>
<tr>
<td>1.1 Workflow Definition versus Workflow Instance</td>
<td>7</td>
</tr>
<tr>
<td>1.2 Conventions, Restrictions, and Limits</td>
<td>7</td>
</tr>
<tr>
<td>1.3 Status Changes for Workflow Instances</td>
<td>10</td>
</tr>
<tr>
<td>1.4 Status Changes for Task Instances</td>
<td>11</td>
</tr>
<tr>
<td>1.5 Supported Languages</td>
<td>12</td>
</tr>
<tr>
<td>1.6 Browser Support</td>
<td>14</td>
</tr>
<tr>
<td><strong>Administration</strong></td>
<td>15</td>
</tr>
<tr>
<td>2.1 Configuring the Workflow Service</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Managing Workflows Using the Monitor Workflows App</td>
<td>23</td>
</tr>
<tr>
<td>2.3 Export Workflow Service Data</td>
<td>30</td>
</tr>
<tr>
<td>2.4 Deactivate the Workflow Service</td>
<td>31</td>
</tr>
<tr>
<td><strong>Development</strong></td>
<td>33</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>33</td>
</tr>
<tr>
<td>3.2 Modeling a Workflow</td>
<td>33</td>
</tr>
<tr>
<td>3.3 Create a Workflow Sample Application</td>
<td>81</td>
</tr>
<tr>
<td>3.4 Creating User Interfaces</td>
<td>82</td>
</tr>
<tr>
<td>3.5 Using Workflow APIs</td>
<td>119</td>
</tr>
</tbody>
</table>
Access Workflow APIs Using OAuth 2.0 Authentication (Client Credentials) .............................................. 120
Access Workflow APIs Using OAuth 2.0 Authentication (Authorization Grant) ........................................... 122
Determine the Service Host. .................................................................................................................. 124
Modifying the Context of a Workflow Instance. ..................................................................................... 124
Updating Task Properties. .................................................................................................................. 127
Workflow Execution Log. .................................................................................................................... 130
Error Codes. ..................................................................................................................................... 131

4 User Guide. ........................................................................................................................................ 133
4.1 Introduction. ..................................................................................................................................... 133
4.2 Working with Tasks in My Inbox. ..................................................................................................... 133
Access the My Inbox Application. ......................................................................................................... 134

5 Security Guide. .................................................................................................................................. 135
5.1 Architecture. ..................................................................................................................................... 135
5.2 Identity Provider and Identity Management. ....................................................................................... 136
5.3 Authorization Configuration. ............................................................................................................ 137
5.4 Destinations. ...................................................................................................................................... 139
  Neo Environment: Configure a Service Task Destination with OAuth2 Client Credentials Flow ............... 140
  Configure a Service Task Destination with OAuth2SAMLBearerAssertion for Principal Propagation. .......... 141
5.5 Data Protection and Data Privacy. ...................................................................................................... 143
  Information Report. .......................................................................................................................... 143
  Erasure. ............................................................................................................................................. 143
  Change Log. ....................................................................................................................................... 144
  Glossary. ......................................................................................................................................... 145

6 Troubleshooting. ................................................................................................................................. 147
6.1 End Users Can't Open SAP Fiori Launchpad Tiles. .............................................................................. 147
  Clear the SAP Fiori Launchpad Cache. ............................................................................................... 148
  Change the SAPUI5 Version. ............................................................................................................... 149
6.2 Error When Clicking "Go to Service" on Portal Tile. ............................................................................ 152
6.3 HTTP Status 403: User Doesn't Have Sufficient Privileges. ............................................................ 153
6.4 Tasks Not Appearing in My Inbox. .................................................................................................... 153
6.5 Error During Workflow Deployment in SAP Web IDE. .................................................................... 154
6.6 No Permissions Granted. ................................................................................................................ 154
6.7 Workflow Service Cannot be Enabled in the Account. ..................................................................... 155
6.8 Category for Web IDE Template Not Available. .............................................................................. 155
6.9 Cannot Deploy Workflow. .............................................................................................................. 155
6.10 Service Calls Fail. ........................................................................................................................ 156
6.11 My Inbox Features Not Active. ....................................................................................................... 157
6.12 Authentication Issues when Using a Custom IDP and Basic Authentication. ............................... 157
1 General

The SAP Cloud Platform Workflow offers modern 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 33]
  Use a Web-based Business Process Model and Notation (BPMN) editor to design workflows with the graphical user interface (UI).

- Using Workflow APIs [page 119]
  Manage workflow instances and workflow definitions using the REST API or the Monitor Workflows app.

- Working with Tasks in My Inbox [page 133]
  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.

Related Information
1.1 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 applications. For more information, see Managing Workflows Using the Monitor Workflows App [page 23].

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.

1.2 Conventions, Restrictions, and Limits

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

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

Note

Limits are, to the extent possible, subject to change.
### Execution Limits

<table>
<thead>
<tr>
<th>Area</th>
<th>Limit</th>
<th>Value</th>
<th>More Information</th>
</tr>
</thead>
</table>
| Workflow context            | Size of the workflow context  | 100 KB per workflow instance | • Creating and Reading Workflow Context Structures [page 58]  
• Applies also if exceeded only temporarily  
• Applies to any operation on the workflow context, that is, to all types of tasks and all types of APIs. |
| API                         | Request rate limit            | 150 requests per second and tenant | • Using Workflow APIs [page 119]  
• Includes requests triggered from user interfaces delivered by SAP  
• In exceptional situations, requests are temporarily rate-limited to a lower value than the given value. |
|                             | Request body size             | 512,000 bytes                | n.a.                                                                                                                                              |
|                             | Processing time               | 30 seconds                   | Includes response generation by the server                                                                                                       |
| Script tasks                | Execution time                | 150 milliseconds             | Configure Script Tasks [page 56]                                                                                                                 |
|                             | Connection timeout            | 1 minute                     | Time to establish the connection with the remote host                                                                                           |
|                             | Socket timeout                | 3 minutes                    | Maximum period between two data packets                                                                                                          |
|                             | Total execution time          | 4 minutes                    | For recommendations on how to implement service tasks if high execution times are common, see Configure Service Tasks [page 51].                   |

### Restrictions

- **Variable Names**  
  There are many ways to create, change, or delete variables in the context of a process. For example, when starting the process, using script tasks, updating the process context manually. In all cases, the names of the variables in the process context must adhere to the following rules:  
  ○ Must not start with "SAP_WFS"  
  ○ Must start with a letter (latin alphabet, or A-Z, a-z)  
  ○ Can contain additional letters, digits, and underscores

- **Duration**  
  When expressions are used to specify duration, they must resolve to ISO 8601 format during runtime. For more information, see ISO 8601.
However, you must consider the following:

- The smallest units which the duration specification supports are minutes.
- The "Week" unit ("W") is not supported.
- The duration specification supports integers only. Also, while using the static mode, the value of the duration field must be less than 2147483647.

**Model Limits**

The following model limits apply to the workflow service.

**Common Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>64</td>
</tr>
<tr>
<td>Documentation</td>
<td>2000</td>
</tr>
</tbody>
</table>

**Workflow Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow</td>
<td></td>
</tr>
<tr>
<td>Subject</td>
<td>255</td>
</tr>
<tr>
<td>Business Key</td>
<td>255</td>
</tr>
</tbody>
</table>

**Flow Element Properties**

**User Task**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>255</td>
</tr>
<tr>
<td>Description</td>
<td>2000</td>
</tr>
<tr>
<td>Users</td>
<td>Maximum of 100 users, maximum of 255 characters per user</td>
</tr>
<tr>
<td>Groups</td>
<td>Maximum of 100 users, maximum of 255 characters per user</td>
</tr>
<tr>
<td>Custom Attribute</td>
<td>15 custom attributes per user task at a time, 30 unique attributes per user task across all workflow versions</td>
</tr>
</tbody>
</table>

**Service Task**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination</td>
<td>200</td>
</tr>
</tbody>
</table>
## 1.3 Status Changes for Workflow Instances

Workflow instances follow a status and action model.

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

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

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

You can move an instance that cannot reach any end event or is no longer required to the CANCELED status. You can also temporarily move an instance to the SUSPENDED status and resume it later. You can change the status using the buttons on the Workflow Instances tile of the Monitor Workflows app. For more information, see Managing Workflows Using the Monitor Workflows App [page 23].

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

Start status of an instance: RUNNING

Final status of an instance: CANCELED, COMPLETED
1.4 Status Changes for Task Instances

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

When a new user task is created without a processor its status is READY. In My Inbox, all users who are listed as recipient users or are assigned to at least one recipient group can see this task.

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

The REST API called from a custom task UI when the user completes a task sets its status to COMPLETED, see Adding Task Completion Buttons [page 87].

A user task has the status CANCELED when a canceling boundary event on the user task triggers it or when the workflow instance of the task is canceled. For more information, see Configure Boundary Timer Events [page 42] and Managing Workflows Using the Monitor Workflows App [page 23].

My Inbox does not display user tasks with status CANCELED or COMPLETED.
1.5 Supported Languages

The workflow service is available in the following languages.

- Workflow documentation:
  - Chinese (Simplified)
  - English
  - Japanese
- Workflow editor: English
- Workflow application: English
- Monitoring user interfaces for workflow administrators:
  - Arabic
  - Chinese (Simplified)
  - Czech
  - Danish
  - Dutch
  - English
  - French
  - German
  - Hebrew
  - Hungarian
  - Italian
  - Japanese
  - Norwegian
  - Polish
- Portuguese (Brazil)
- Russian
- Spanish
- Turkish
- Inbox application for workflow participants:
  - Arabic
  - Bulgarian
  - Catalan
  - Chinese
  - Chinese trad.
  - Croatian
  - Czech
  - Danish
  - Dutch
  - English
  - Estonian
  - Finnish
  - French
  - German
  - Greek
  - Hebrew
  - Hindi
  - Hungarian
  - Italian
  - Japanese
  - Kazakh
  - Korean
  - Latvian
  - Lithuanian
  - Malay
  - Norwegian
  - Polish
  - Portuguese
  - Romanian
  - Russian
  - Serbian (Latin)
  - Slovak
  - Slovenian
  - Spanish
  - Swedish
  - Thai
  - Turkish
  - Ukrainian
  - Vietnamese
To activate the translations for each required language in SAP Fiori launchpad, see Working with Business Content.

### 1.6 Browser Support

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

**i Note**

The workflow editor does not support the Safari browser.

**Supported Browsers**

<table>
<thead>
<tr>
<th>Browser</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Internet Explorer</td>
<td>11</td>
</tr>
<tr>
<td>Mozilla Firefox</td>
<td>Extended Support Release (ESR) and latest version</td>
</tr>
<tr>
<td>Google Chrome</td>
<td>Latest version</td>
</tr>
<tr>
<td>Safari</td>
<td>7.0 and upwards (for Mac OS X only)</td>
</tr>
</tbody>
</table>
2 Administration

Configuration tasks for the SAP Cloud Platform Workflow service.

Related Information

Configuring the Workflow Service [page 15]
Managing Workflows Using the Monitor Workflows App [page 23]
Export Workflow Service Data [page 30]

2.1 Configuring 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, SAP Web IDE Full-Stack, 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.
   d. Go back to Services and search and enable the SAP Web IDE Full-Stack.
   e. Go back to Services and 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 139].
1. Creates the two destinations `bpmworkfournitetime` and `iwsworkspacetruntime`. Please do not change them.

   **Note**

   Both configurations are required to run the workflow service.

2. 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 137].

   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.

3. Configure SAP Fiori launchpad for My Inbox.

   For more information, see Configuring SAP Fiori Launchpad Objects [page 16].

**Related Information**

SAP Cloud Platform Portal documentation

**2.1.1 Configuring 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 18].
  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 Add content to the page Typical Workflow of an Administrator 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.
2.1.1.1  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.

2.1.1.2  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 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 launchpad configuration cockpit and maintaining the respective tcmURL parameter.

Note

Please note that the URL parameter name is case sensitive.

Example

![Image of My Inbox configuration with TCM parameters](image-url)
You can expose this application as an SAP Fiori launchpad tile, which launches a separate instance of My Inbox application connected to the configured TCM-compliant OData service.

### 2.1.2 Configure the Workflow Service Mail Destination

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

#### Prerequisites

- You know the necessary details for SMTP mail configuration of your scenario.
- Your mail server has the following characteristics:
  - It supports the SMTP STARTTLS command on ports 587 or 465, because the workflow service supports only STARTTLS on these ports.
  - It requires authentication, because the workflow service does not support unauthenticated logon.

#### Procedure

1. In the SAP Cloud Platform cockpit, configure the workflow service.
   
   a. In the navigation area, choose *Services*.
   
   b. Search for *Workflow*.
   
   c. On the *Workflow* tile, choose *Configure Service*.

2. Create a new destination, or use the template file.
   
   ○ *Import Destination*
   
   1. Save the template as a file:

```plaintext
Type=MAIL
Name=bpmworkflowruntime_mail
mail.user=
mail.password=
mail.smtp.host=mail.example.com
mail.smtp.port=587
mail.transport.protocol=smtp
mail.smtp.starttls.required=true
mail.smtp.starttls.enable=true
mail.smtp.auth=true
mail.smtp.from=cpworkflow@example.com
mail.smtp.ssl.checkserveridentity=true
mail.bpm.send.disabled=false
```

   2. Import the destination from the file, and set the values for user, password, host, port, and from address.
**New Destination**

Enter the following data:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>bpmworkflowruntime_mail</td>
</tr>
<tr>
<td>Type</td>
<td>Mail</td>
</tr>
<tr>
<td>Description</td>
<td>Text, for example, <em>Workflow service mail destination</em></td>
</tr>
<tr>
<td>User</td>
<td>User for logging on to the mail server</td>
</tr>
<tr>
<td>Password</td>
<td>Password for logging on to the mail server</td>
</tr>
</tbody>
</table>

Use *New Property* to add the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail.transport.protocol</td>
<td>smtp</td>
</tr>
<tr>
<td>mail.smtp.auth</td>
<td>true</td>
</tr>
<tr>
<td>mail.smtp.starttls.required</td>
<td>true</td>
</tr>
<tr>
<td>mail.smtp.host</td>
<td>Host name of your mail server</td>
</tr>
<tr>
<td>mail.smtp.port</td>
<td>Port where your mail servers listens for connections (typically 587, in rare cases 465)</td>
</tr>
<tr>
<td>mail.smtp.from</td>
<td>Mail address to use as the “From” address of mails sent by the workflow service, for example, <a href="mailto:cpworkflow@example.com">cpworkflow@example.com</a> This address must belong to an existing mailbox because it receives the replies to mails that the workflow service sends.</td>
</tr>
<tr>
<td>mail.smtp.ssl.checkserveridentity</td>
<td>Optional: <em>true</em> or <em>false</em>; default is <em>true</em> if no value is provided</td>
</tr>
<tr>
<td>mail.smtp.ssl.trust</td>
<td>* or a space-separated list of acceptable host names</td>
</tr>
<tr>
<td>mail.bpm.send.disabled</td>
<td>o <em>true</em> Tiny interaction with the mail server, for example, temporarily while you develop a workflow o <em>false</em></td>
</tr>
</tbody>
</table>
2.1.3 Configuring Principal Propagation for Service Tasks

When starting a workflow or completing a task, you can use principal propagation to forward the information about who is logged on to the services that are called from a workflow instance.

Before you can use principal propagation, the following one-time configurations are required:

- An OAuth 2.0 client with Authorization Code Grant. For more information, see Create an OAuth 2.0 Client with Authorization Code Grant [page 21].
- A destination that uses the credentials of that OAuth client. For more information, see Create an OAuth Destination [page 22].

You need one destination for each service that is called. For more information, see Configure Service Tasks [page 51] and Destinations [page 139].

2.1.3.1 Create an OAuth 2.0 Client with Authorization Code Grant

Procedure

1. Register a new OAuth client as described in OAuth 2.0 Configuration.
2. Make note of the Authorization Endpoint and the Token Endpoint URLs, as you’ll need them later for other configurations.
3. Specify the client configuration as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a name, for example Cloud Platform Workflow OAuth Client for Principal Propagation.</td>
</tr>
<tr>
<td>Subscription</td>
<td>Choose your SAP Cloud Platform Workflow subscription. That is, the entry that ends with bpmworkflowruntime.</td>
</tr>
<tr>
<td>Field</td>
<td>Comment</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ID</strong></td>
<td>Use the client ID that’s automatically assigned, or regenerate a new one. Make note of this ID, as you’ll need it for other configurations.</td>
</tr>
<tr>
<td><strong>Authorization Grant</strong></td>
<td><strong>Authorization Code</strong></td>
</tr>
<tr>
<td><strong>Confidential</strong></td>
<td>Select this option.</td>
</tr>
<tr>
<td><strong>Secret</strong></td>
<td>Enter the secret, for example, a password. Make note of this, as you’ll need it for other configurations.</td>
</tr>
<tr>
<td><strong>Skip Consent Screen</strong></td>
<td>Select this option.</td>
</tr>
<tr>
<td><strong>Redirect URI</strong></td>
<td>Enter the service host, ending in /workflow-service.</td>
</tr>
<tr>
<td></td>
<td>Example: bpmworkflowruntime&lt;providerid&gt;-&lt;subscriberaccountid&gt;.&lt;region&gt;.hana.ondemand.com/workflow-service</td>
</tr>
<tr>
<td></td>
<td>For more information, see Determine the Service Host [page 124].</td>
</tr>
<tr>
<td><strong>Token Lifetime</strong></td>
<td>5 minutes</td>
</tr>
<tr>
<td><strong>Refresh Token Lifetime</strong></td>
<td>Leave empty for unlimited lifetime.</td>
</tr>
</tbody>
</table>

### 2.1.3.2 Create an OAuth Destination

**Prerequisites**

Log in to the cockpit and open the Destinations editor.

**Procedure**

1. Create an OAuth destination for the SAP Cloud Platform Workflow runtime as described in Create HTTP Destinations.
2. Choose New Destination.
3. Enter the destination name bpmworkflowruntimeoauth.
4. Specify the Type as HTTP.
5. Enter Cloud Platform Workflow OAuth Destination for Principal Propagation as the description.
6. Select a Proxy Type of Internet.
7. Select Basic Authentication for the connection.
8. Set the destination URL to the authorization endpoint URL from (Create an OAuth 2.0 Client with Authorization Code Grant [page 21]) and remove the path parts. Enter, for example, a URL that is similar to https://oauthasservices-<accountid>.<region>.hana.ondemand.com.
9. Set the user to the client ID from Create an OAuth 2.0 Client with Authorization Code Grant [page 21].
10. Set the password to the secret from Create an OAuth 2.0 Client with Authorization Code Grant [page 21].

2.2 Managing Workflows Using the Monitor Workflows App

With the web-based administration Monitor Workflows app you can manage workflow instances and workflow definitions.

The app offers two interlinked views, one for workflow instances and one for workflow definitions. Both can be accessed using dedicated tiles in the SAP Fiori launchpad.

Note
You must have the latest maintenance version or latest version of SAP UI5 configured on SAP Fiori launchpad to use the Monitor Workflows app.

Related Information

Managing Workflow Instances [page 24]
Managing Task Instances [page 25]
Managing Workflow Definitions [page 27]
Deep Linking in Monitor Workflows App [page 29]
2.2.1 Managing Workflow Instances

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

Prerequisites

The SAP Fiori launchpad objects are configured. For more information, see Configuring SAP Fiori Launchpad Objects [page 16].
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 search the workflow instances**, type the keyword you want to use in the Search field, and choose \( \text{Search} \), or choose Enter.
- **To filter for workflow instances based on status and definition**, choose \( \text{Filter} \) in the workflow instance list.

\[ i \text{ Note} \]
By default, the filter is applied to show workflow instances in running, erroneous and suspended status. However, you can also filter workflow instances that are in completed and canceled status.

- **To display details about a workflow instance and to navigate to it**, select a workflow instance.
- **To view the current context of a workflow instance**, choose the Workflow Context tab.
- **To view tasks of a workflow instance**, choose Show Tasks on the details screen of the 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.
  You can also view the count of instances displayed in the view versus the total instance count.
- **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.

**Related Information**

- Managing Workflows Using the Monitor Workflows App [page 23]
- Deep Linking in Monitor Workflows App [page 29]

### 2.2.2 Managing Task Instances

The task instances view shows tasks for a given workflow instance.

**Prerequisites**

The SAP Fiori launchpad objects are configured. For more information, see Configuring SAP Fiori Launchpad Objects [page 16].
The following actions are available:

- To display details about a task instance, select the task.
- To navigate to the workflow instance of the task, click the workflow instance ID on the details screen.
- To navigate to the workflow definition, click the workflow definition ID on the details screen.
- To assign a processor for a task, choose Assign Processor. This action is only available for tasks with status Ready or Reserved. When you assign a task to a user, the user becomes its processor. All other task recipients can no longer see the task in My Inbox. The status of the task changes from Ready to Reserved.
- To unassign the processor of a task, choose Unassign Processor. This action is only available for tasks with status Reserved. When you unassign a task from a processor, it is available again to all recipients and they can see the task in My Inbox. The status of the task changes from Reserved to Ready.

Related Information

Managing Workflows Using the Monitor Workflows App [page 23]
Deep Linking in Monitor Workflows App [page 29]
2.2.3 Managing Workflow Definitions

The workflow definitions view shows a list of deployed workflow definitions.

Prerequisites

The SAP Fiori launchpad objects are configured. For more information, see Configuring SAP Fiori Launchpad Objects [page 16].

The following actions are available:

- To filter the workflow definitions, use the following criteria: workflow definition ID, workflow definition name, or the workflow definition version.
- To search the workflow definitions, type the keyword you want to use in the Search field, and choose (Search), or press Enter.
- To start a new workflow instance, select a workflow definition and choose Start New Instance. If you have configured a sample context while modeling a start event, it is shown as the context data while starting a new workflow instance in the Start New Instance window. However, you can also modify this JSON context data as required. For more information, see Configure Start Events [page 68].
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 load more workflow definitions, scroll down to the end of the list and choose More.
- To download the workflow model, select the definition from the list, then choose Download Workflow Model. With this you retrieve the workflow model for the latest deployed version of a workflow definition.

**Note**
You are recommended not to import this downloaded workflow model to SAP Web IDE Full-Stack.

**Related Information**

Managing Workflows Using the Monitor Workflows App [page 23]
Deep Linking in Monitor Workflows App [page 29]
2.2.4 Deep Linking in Monitor Workflows App

You can access the workflow definitions, instances, and task instances using direct URLs. You can use the below URL formats to access the required information.

**Workflow Instance**

- To access the list of workflow instances, use the following URL format:
  
  https://flpsandbox-<consumer_account>.<landscape_host>/sites?
  siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances

- To access a particular workflow instance, use the following URL format:
  
  https://flpsandbox-<consumer_account>.<landscape_host>/sites?
  siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances/<workflow_instance_id>

**Workflow Definition**

- To access the list of workflow definitions, use the following URL format:
  
  https://flpsandbox-<consumer_account>.<landscape_host>/sites?
  siteId=<site_id>#bpmworkflowmonitor-DisplayDefinitions&/workflowDefinitions

- To access a particular workflow definition, use the following URL format:
  
  https://flpsandbox-<consumer_account>.<landscape_host>/sites?
  siteId=<site_id>#bpmworkflowmonitor-DisplayDefinitions&/workflowDefinitions/<workflow_definition_id>

**Task Instance**

- To access the list of task instances for a particular workflow instance, use the following URL format:
  
  https://flpsandbox-<consumer_account>.<landscape_host>/sites?
  siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances/<workflow_instance_id>/taskInstances

- To access a particular task instance, use the following URL format:
  
  https://flpsandbox-<consumer_account>.<landscape_host>/sites?
  siteId=<site_id>#bpmworkflowmonitor-DisplayInstances&/workflowInstances/<workflow_instance_id>/taskInstances/<task_instance_id>

**Note**

If you are using the default site, then site_id in the URL is not mandatory.
2.3 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, form definitions, and workflow instances.

Context

⚠️ Caution

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

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

- Design time or modeling artifacts from the workflow editor.
  For more information, see Transport Workflows between Accounts [page 74].
- Runtime artifacts from the workflow service using the workflow service API.
  You export data related to a workflow definition, form 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 124].
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 form-definitions.json file that contains a list of the latest deployed form definitions.
- A workflow-definitions.json file that contains a list of the latest deployed workflow definitions.
- A workflow-instances.json file that contains a list of all workflow instances available on the system.
- 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.
- A form-definition-data folder: For each form definition on the system one file (<form-definition-ID>.json) is written. It contains form definition metadata of all versions deployed.

2.4 Deactivate the Workflow Service

Administrators of the SAP Cloud Platform account can disable the SAP Cloud Platform Workflow.

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.

Context

⚠ Caution
If you deactivate your workflow service, you also delete all data from the database, the subscription, and the database bindings.
Currently, deactivating the service does not delete:

- The assignment between users and workflow service roles
- HTTP destinations
- Portal content

**Note**
If you later reenable the workflow service, the items that were not deleted become active again.

**Procedure**

In the SAP Cloud Platform cockpit, disable the SAP Cloud Platform Workflow service for your subaccount.

a. In the navigation area, choose Services.
b. Search for SAP Cloud Platform Workflow.
c. On the Workflow tile, choose Disable.


3 Development

Introduction [page 33]
General [page 5]
Modeling a Workflow [page 33]
Defining a User Interface for a Workflow [page 82]
Using Workflow APIs [page 119]

3.1 Introduction

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

Related Information

Modeling a Workflow [page 33]
Defining a User Interface for a Workflow [page 82]
Using Workflow APIs [page 119]

3.2 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 Neo regions where the SAP Cloud Platform Workflow service is offered.
- The workflow editor does not support Safari browser.

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 68].
• 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 39].
  ○ Gateways: Gateways control the flow of execution in a workflow. For more information, see Gateways [page 71].
• 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 68].

Related Information

Enable the Workflow Editor in SAP Web IDE [page 34]
Create a New Workflow Project [page 35]
Deploy Workflows [page 75]
Accelerated Modeling with Speed Buttons [page 76]
Expressions [page 77]

3.2.1 Enable the Workflow Editor in SAP Web IDE

In the Neo environment, you must enable the workflow editor feature to model workflows in SAP Web IDE Full-Stack.

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.
3.2.2 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 Full-Stack application.
2. From the left pane, choose (Development) and navigate to the Workspace folder.
3. Choose File > New > Project from Template.
4. From the Template Selection screen, choose Category as Business Process Management.
5. Choose the Workflow Project tile, then choose Next.
6. On the Basic Information screen, enter a project name, then choose Next.
7. In the Workflow Details screen, provide a workflow name and optional description.

**i Note**

To create multiple workflows, you can select a workflow project or the workflow folder and choose New > 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 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.
3.2.2.1 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.

3.2.2.2 Editor Layout

The workflow editor consists of the following areas:

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

You use this procedure to define workflows.

Procedure

1. Open the workflow with the Workflow Editor.
   
   For more information, see Open Workflow Files in the Workflow Editor [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 77].

   i Note

   ○ For more information on character limits for workflow service, see Conventions, Restrictions, and Limits [page 7].

   ○ 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 7].

   ○ 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 119].

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.

   i 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.

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

6. Optional: Configure a sample context while modeling a start event. After the deployment of the workflow, the sample context is displayed in the **Monitor Workflows** app while starting a new workflow instance. For more information, see Configure Start Events [page 68].
   
   You can also retrieve the configured sample context using the Public API. For more information, see SAP Cloud Platform Workflow API.

7. To add a task to the workflow, see **Tasks** [page 39].

8. To add a gateway to the workflow, see **Gateways** [page 71].

9. To add an intermediate message event, see Configure Intermediate Message Events [page 69].

10. To add an intermediate timer event, see Configure Intermediate Timer Events [page 71].

11. To connect two flow elements, choose the **i 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. 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.

    For more information about speed buttons, see Accelerated Modeling with Speed Buttons [page 76].

12. To model the end event of a workflow, choose **Events** ➤ **End Event** and drop it onto the canvas from the palette.

13. In the **End Event Properties** icon from the first flow pane, provide a name and documentation for the end event.

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

    For more information on the terminate end event, see **Events** [page 68].

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

15. In the **Terminate End Event Properties Properties** pane, provide a name and description for terminate end event.

16. To format the workflow model, choose **Arrange Horizontally** or **Arrange Vertically** from the toolbar.

17. Choose **Save**.

   → **Recommendation**

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

   ○ Each time you make changes to the properties of flow elements, ensure that you press the **ENTER** key.
3.2.3.1 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 the processor of the task can complete the task instance, and view the 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 39]
Configure Service Tasks [page 51]
Configure Script Tasks [page 56]

3.2.3.1.1 Configure User Tasks

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

Prerequisites

To ensure that end users can view tasks in custom UIs in My Inbox, the following configuration steps are required:

- 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.

You can propagate the user who completes a task to a service called later by a service task in the same workflow instance. For more information, see Configure Service Tasks [page 51].

Procedure

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

   Note
   - For more information on character limits for workflow service, see Conventions, Restrictions, and Limits [page 7].
   - 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 a 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

   Note
   Priority is reflected in My Inbox using which the end user can sort, filter, and group the tasks. For more information, see Working with Tasks in My Inbox [page 133].

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.

**i Note**
- **Subject**, **Description**, **Users**, and **Groups** can also refer to the dynamic workflow context. For example, if you want to provide a **Subject** that references a variable from dynamic context, you can specify the expression in **Subject** field as "Approval for ${context.employee.name}". For more information, see Expressions [page 77].
- For users and groups, either use a context reference that resolves to a string with different users or groups separated by commas or a context reference that resolves to an array of strings.
- To provide multiple users or groups of users to process the task, separate each unique ID with a comma.
- You can assign a maximum number of 100 users or groups as recipients to a user task.
- Recipients can view these tasks in My Inbox. They can also complete these tasks, which further proceeds the workflow execution.

10. To configure the duration by when the task is due, select the **Configure Due Date** checkbox.

You must configure the due date using the following substeps:

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

**i Note**
You must provide an expression in the **Duration** field using a subset of the ISO 8601 format. For example, **PT${context.minutes}M**. The JUEL expression ${context.minutes} is evaluated at runtime. You can provide multiple duration attributes by using multiple JUEL expressions. For more information about the duration formats that are supported in ISO 8601, see Conventions, Restrictions, and Limits [page 7].

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

**i 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 133].

11. Configure a custom task user interface.

You have the following options:

- Configure a Custom Task User Interface Using an HTML5 App [page 44]
- Configure a Task User Interface Using Workflow Forms [page 47]

12. Assign custom attributes to a user task. For more information, see Configure Custom Attributes [page 48].

13. (Optional) To include a timer for the user task, add a boundary timer event. For more information, see Configure Boundary Timer Events [page 42].

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

15. Choose **Save**.
3.2.3.1.1 Configure Boundary Timer Events

Configure a boundary timer event to trigger an alternative flow if a user task doesn’t finish within a specified time 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 \textit{Duration} field. You can use one of the following ways to configure this field:

- To use expressions, choose \textit{Expression} from the \textit{Duration Based On} dropdown.

  \textbf{i Note}
  
  You must provide an expression in the \textit{Duration} field using a subset of the ISO 8601. For example, \texttt{PT$(\text{context.minutes})M$}. The JUEL expression \texttt{$\{\text{context.minutes}\}$} is evaluated at runtime. You can provide multiple duration attributes by using multiple JUEL expressions. For more information about the duration formats that are supported in ISO 8601, see Conventions, Restrictions, and Limits [page 7].

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

- To use the due date value as the duration, choose \textit{Task Due Date} from the \textit{Duration Based On} dropdown.

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

5. To define the boundary timer event as canceling, select the \textit{Cancel Task} checkbox.

6. Choose \textit{Save}.

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

\subsection*{3.2.3.1.1.2 Create a Basic Task User Interface for a User Task}

You can create a basic task user interface that can be customized for your use case.

\textbf{Context}

After customizing the basic task UI, you can deploy the same and use it while configuring a user task.
Procedure

1. Log in to the SAP Web IDE Full-Stack application.
2. From the left pane, choose \( \text{Development} \).
3. In the context menu of the required workflow project, choose \( \text{New} \ > \text{Workflow Task UI} \).
4. In the Component Information screen, provide a name for the SAPUI5 component.

   **i Note**
   This is the folder name for a particular task UI.

5. Choose Next.
6. In the confirmation tab, choose Finish.

Results

A new folder is created under the workflow project, which contains the generated SAPUI5 Component. You can modify this as per your requirement.

### 3.2.3.1.1.3 Configure a Custom Task User Interface Using an HTML5 App

**Procedure**

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

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

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

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

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML5 App Name</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>&lt;.component&gt;</code> suffix</td>
</tr>
</tbody>
</table>

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

- Grunt build is not enabled in SAP Web IDE Full-Stack or SAPUI5 Client Build is not enabled on SAP Web IDE.

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

  ![Component.js](image)

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

- Grunt build is enabled in SAP Web IDE Full-Stack or SAPUI5 Client Build is enabled on SAP Web IDE. For more information, see [Building Applications Using Grunt](#) and [Building Applications Using SAPUI5 Build](#).

  Configuration of the above *User Interface* properties based on this scenario is illustrated with an example:
3. In the Parameters field, provide the configuration data that can be accessed at runtime.

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

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

For example, a task UI contains three actions namely: Accept, Reject, Rework. If Rework action is not required for some specific tasks, you can still reuse this UI. This can be done by passing some parameters as configuration data. The task UI developer can then access these parameters in his/her custom task UI and choose to show or hide specific actions. For more information, see Retrieve Parameters in Custom Task UI [page 91].

**i Note**

- You can enter multiple key value pairs separated by comma. If the value contains a comma, then you can use backslash as shown below:
  
  `key1=value1, key2=value2\, value3\, value4`

- The key values in this case are as follows:
  
  `key1=value1`
  
  `key2=value2, value3, value4`

- The following points must be considered while providing keys:
  
  - Keys cannot contain JUEL expressions and must be static.
Keys must start with a letter and can contain only alphanumeric characters.
- Keys cannot contain whitespaces or special characters with an exception of underscores.
- Values can be static or can contain JUEL expressions. For more information, see Expressions [page 77].
- Expressions are evaluated at task creation time, that is, they cannot be used to transfer data to the user interface that changes after this time. For such cases, refer to Read Task Context Data [page 86].

4. Save the workflow.

Related Information

Configure User Tasks [page 39]

3.2.3.1.1.4 Configure a Task User Interface Using Workflow Forms

Procedure

1. In the workflow editor, select the user task and choose User Interface.
2. Under Type, choose Form.
3. Under Form Details, choose one of the following options:
   - Create File
     On the New Form dialog, enter the following data:
     
     | Field    | Description                      |
     |----------|----------------------------------|
     | Name     | Name of the form you create      |
     | ID       | Identifier of the new form       |
     | Revision | Revision of the form             |
     |          | For more information, see Version Forms [page 116]. |
   - Select
     On the Select Form dialog, enter the following data:
### Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Name</strong></td>
<td>Name of the project. The name is preset to the current project and you cannot change it.</td>
</tr>
<tr>
<td><strong>File Name</strong></td>
<td>Name of the form from the list of forms that are available in the current project only</td>
</tr>
<tr>
<td><strong>Form Revision</strong></td>
<td>Revision of the form For more information, see <a href="#">Version Forms</a></td>
</tr>
</tbody>
</table>

4. Save the workflow.

The form is created in a separate `forms` folder within the workflow project in a folder named exactly like the workflow for which the form is created. Same as for script tasks.

**Related Information**

[Configure User Tasks](#) [page 39]

### 3.2.3.1.1.5 Configure Custom Attributes

You can assign custom attributes to user tasks.

**Context**

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

At runtime, you can use the Workflow Service API or the Task Consumption Model API to search for custom attributes or to find the respective task instances. For more information about the characteristics of the various APIs, see [Using Workflow APIs](#) [page 119].

**Procedure**

1. Choose the Attributes tab.
2. To add a new row, choose Add.
3. Provide the following details in the table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>A unique identifier of the attribute within a user task.</td>
</tr>
<tr>
<td></td>
<td><strong>i Note</strong></td>
</tr>
<tr>
<td></td>
<td>You can only use alphanumeric characters as ID and it must only start with an alphabet.</td>
</tr>
<tr>
<td>Label</td>
<td>A human readable name of the attribute, which can be used by appropriate user interfaces to label the attribute.</td>
</tr>
<tr>
<td>Type</td>
<td>Data type of the attribute.</td>
</tr>
<tr>
<td>Value</td>
<td>It can be a constant or an expression, which gets resolved upon task creation.</td>
</tr>
</tbody>
</table>

**i Note**

A user task can contain up to 15 attributes at a time. For more information, see Conventions, Restrictions, and Limits [page 7].

4. Save the changes.

**Related Information**

Configure User Tasks [page 39]
Display Custom Attributes in My Inbox [page 49]

3.2.3.1.5.1 Display Custom Attributes in My Inbox

It is possible to use custom attributes in *My Inbox*. Those can be custom attributes with predefined names or other custom attributes assigned to your tasks.

**Context**

This feature allows you to replace the task title, display a KPI indicator, display a KPI unit, and display an additional custom attribute.

The custom attributes that are visible in the *Task List*, are also displayed in the *Details View* header data.
The predefined custom attributes are `CustomTaskTitle`, `CustomNumberValue`, `CustomNumberUnitValue`, and `CustomObjectAttributeValue`.

**Procedure**

1. Name the custom attributes as follows:
   - `CustomTaskTitle`
   - `CustomNumberValue`
   - `CustomNumberUnitValue`
   - `CustomObjectAttributeValue`

My Inbox maps the predefined names to the specific locations in the **Task List**.

The attributes, which are shown in the **Task List** are not displayed in the description tab of the **Master-Detail View** of the selected task.

**Example**

![Example Image]

**i Note**

Please, note that by default this feature is disabled in **My Inbox**. To enable it, the administrator has to configure the additional parameter `showAdditionalAttributes=true` in the app configuration of **My Inbox**.
2. (Optional) To assign other custom attributes to your tasks, see Configure Custom Attributes [page 48].

In My Inbox, those custom attributes are visualized as part of the description for the task.

![Example](image)

**Example**

In My Inbox, those custom attributes are visualized as part of the description for the task.

**Note**

You can define up to 15 other custom attributes per task, which are displayed in the information tab of a standard task UI.

### 3.2.3.1.2 Configure Service Tasks

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

**Context**

The execution of service tasks is subject to resource limits, for example, with respect to network timeouts. If the target service does not comply with the time restrictions described in Conventions, Restrictions, and Limits [page 7], the connection with the target service is aborted, the service task fails, and the workflow instance is put into the ERRONEOUS state.
Long execution times negatively impact the execution of other tasks of a specific tenant, because there is only a limited number of parallel executions allowed for a tenant. The resource limits enforced by the workflow service therefore have the purpose of freeing up resources as early as possible for other tasks.

→ Tip

We recommend that the service execution time is much less than the limits documented in Conventions, Restrictions, and Limits [page 7]. If high execution times are common, consider building an intermediate service that initiates asynchronous processing of the actual service call and returns quickly. It can report back the execution result with the help of Configure Intermediate Message Events [page 69].

Procedure

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

   i Note

   A unique read-only ID is generated for every workflow artifact.

5. In the Service Task Properties area, choose the Details tab.
6. Provide the Destination.

   i Note

   ○ The destination you provide here is the destination specified in the consumer subaccount, which determines the host to connect to at runtime. For more information about the supported feature set of destinations, see Destinations [page 139].
   ○ For more information on character limits for SAP Cloud Platform Workflow, see Conventions, Restrictions, and Limits [page 7].

7. Select one of the following options from the Choose a Service From list:
   ○ SAP API Business Hub
   ○ Others (default)

   i Note

   SAP API Business Hub is the central catalog, hosted by SAP to discover, explore, and test the SAP and partner APIs that are required to build extensions, or process integrations using SAP Cloud Platform. For more information, see SAP API Business Hub.

8. If you have chosen SAP API Business Hub, perform the following procedure Configure a Service from SAP API Business Hub [page 55].

9. If you selected Others, provide the following details:
   a. Path: Resource path that appends to the URL of the specified destination while calling the service.
i Note

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

b. **HTTP Method**: Specify one of the following HTTP methods: GET, POST, PATCH, PUT, or DELETE.

i Note

If the HTTP method is POST, DELETE, PATCH, or PUT, then the Path to XSRF Token field appears. XSRF token is used for modifying operations that are protected against XSRF (cross-site request forgery) attacks. For more information, see SAP Cloud Platform.

c. **Path to XSRF Token**: The resource path that must be appended to a specified destination, while calling the service to fetch an XSRF token.

d. **Request Variable**: Link to a workflow context node that populates the body of the HTTP request.

i Note

○ The referenced node is used 1:1 as content for the request body.
○ The complete context can be referenced in the request body as follows: `${context}`.
○ If the request variable contains a primitive JSON type (number or string) or literal (null, true, or false), the service must accept an HTTP body following RFC 8259 instead of the older RFC 4627.
○ If the HTTP method is POST, PATCH, or PUT, then you see the Path to XSRF Token field.

e. **Response Variable**: Link to a workflow context node that is created or overwritten to finally store the body of the HTTP response.

i Note

○ The referenced node stores the response body.
○ The complete context cannot be overwritten by the contents of the response body. As a result, the expression `{{context}}` cannot be used in the response variable. A variable within the context must be specified to be used as a response variable. For example, `{{context.leaveRequest}}` or `{{context.leaveRequest.response}}` are valid response variables.

❖ Example

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

The service URL for this example is `https://{host}/leaverequest`. 

SAP Cloud Platform Workflow Development

PUBLIC

53
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.

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

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

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

10. Enable principal propagation.
   a. Select Principal Propagation for the service task.
      For more information, see Configuring Principal Propagation for Service Tasks [page 21].
   b. In the Flow Element section, choose Select.

   **Note**
   This field is available only if principal propagation is active. Then, it is a mandatory field.
   c. To search for the start event or a user task in the workflow, use Select Flow Element.
To propagate the user who started the workflow instance, browse for the start event in the same workflow model.

To propagate the user who completed a user task instance, browse for the user task in the same workflow model.

If a user task is located in a loop, the last completion action of a corresponding task instance in a workflow instance defines the actual user that is propagated.

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

Next Steps

In the Neo environment, 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 Destinations [page 139].

Related Information

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

3.2.3.1.2.1 Configure a Service from SAP API Business Hub

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

Procedure

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

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

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

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


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

### 3.2.3.1.3 Configure Script Tasks

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

**Context**

**i Note**
If you have previously modeled a 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.

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

**Procedure**

1. Choose (Tasks), then Script Task from the palette and drop it on to the canvas.
2. In the Script Task Properties area, provide a name and documentation (optional) for the script task.
3. To add JavaScript files, perform one of the following steps:
   ○ Choose Select to browse for the JavaScript file in the current project.
   ○ To create a JavaScript file, perform the following steps:
     1. Choose the Create File link.
     2. In the Create New File window, provide a file name.
     3. Choose Create.
     4. In the JavaScript file, provide the script.

   **Note**
   ○ You can view and or edit the JavaScript file by selecting the Script File link.
   ○ You can find the JavaScript file in the following location: `<workflow-project>/scripts/<workflow-name>/<script-file-name>.js`.
   ○ For more information about Code Editor, see Developing Applications.
   ○ The provided APIs, as well as the objects and arrays stored in the workflow context, are non-native JavaScript objects; that is, ECMAScript host objects. Their behavior might differ from that of the native objects. For more information about supported APIs, see:
     ○ Creating and Reading Workflow Context Structures [page 58]
     ○ Accessing Contextual Information During Execution of Script Tasks [page 61]
   ○ The script must be in JavaScript that is based on ECMAScript 5.1. For more information, see the Ecma Web page #. Restrictions: 'eval' and 'Function' are not supported for script tasks.
   ○ Using the function keyword is supported, but you cannot assign functions to workflow context variables.
   ○ The execution of script tasks is subject to resource limits, for example, with respect to processing time or memory usage. The limits enforced by the workflow service have the purpose of freeing up resources as early as possible for other tasks. The limits protect against excessive usage, for example, caused by in-efficient programming or unexpected input sizes. If the limits are exceeded, the corresponding workflow instance is put into the ERRONEOUS state. The error is written to the error logs of the workflow instance. You can retrieve the error logs using the REST API or the Monitor Workflows app. If your scripts reach the resource limits, analyze the reasons, for example, large input data. Try to reduce the input size or the complexity of the transformations executed on it.
   ○ For the specific limits that apply to script tasks, see Conventions, Restrictions, and Limits [page 7].

4. Save the workflow.

**Related Information**

- Accelerated Modeling with Speed Buttons [page 76]
- Transport Workflows between Accounts [page 74]
- Conventions, Restrictions, and Limits [page 7]
### 3.2.3.1.3.1 Creating and Reading Workflow Context Structures

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

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

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

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

**i Note**

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

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

#### Reading Variables

```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

// 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 59]
Modifying the Workflow Context with Arrays [page 60]

3.2.3.1.3.11 Modifying the Workflow Context with Objects

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

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

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

Object Property Access

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

Object Conversions

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

3.2.3.1.3.1.2 Modifying the Workflow Context with Arrays

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

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

Constructing Array

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

Manipulating Array

// Insert entries into array at specific positions
var array = $.context.myArray;
if (array.length == 0) {
array.push("first"); // adds a new element at the end of the array
array.splice(1,1); // removes entry at position 1, the one that was previously the first
array.unshift("new first"); // adds a new element at the beginning of the array
// array.splice(-1, 1); // out of bounds deletions
// array.splice(42, 1); // resp. at the last position
}
if (array.length == 1) {
var el = array.shift(); // returns the first element of the array and deletes it from the array
array.unshift("new first"); // adds a new element at the beginning of the array
} var idx = array.indexOf("new first"); // returns the index of the first occurrence of the passed value
// all JavaScript ECMA 5.1 array functions are supported (http://ecma-international.org/ecma-262/5.1/)

Array Index Access

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

3.2.3.1.3.2 Accessing Contextual Information During Execution of Script Tasks

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

Getting Information About the Environment

var workflowInstanceId = $.info.workflowInstanceId; // for example, "336963b0-3726-49fa-bf0c-87a8f7aaaf8"
var workflowDefinitionId= $.info.workflowDefinitionId; // for example, "scripttaskprocess"
var startedByUserId = $.info.startedBy; // for example, "John"
Getting Information About User Task Instances

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

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

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type in Script Task</th>
<th>Task Status READY</th>
<th>Task Status RESERVED</th>
<th>Task Status COMPLETED</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>createdAt</td>
<td>Date*</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>createdBy</td>
<td>String</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>priority</td>
<td>String</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>dueDate</td>
<td>Date*</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>status</td>
<td>String</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>subject</td>
<td>String</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>recipientUsers</td>
<td>Array of strings</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>recipientGroups</td>
<td>Array of strings</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>processor</td>
<td>String</td>
<td>null</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>claimedAt</td>
<td>Date*</td>
<td>null</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>completedAt</td>
<td>Date*</td>
<td>null</td>
<td>null</td>
<td>Available</td>
</tr>
</tbody>
</table>

* Please note that dates are represented as strings in expressions. For more information, see Expressions [page 77].

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

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

Script tasks cannot modify the $.usertasks API. All its properties are provided by SAP Cloud Platform Workflow and are read-only.
Saving Contextual Information in the Workflow Context

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

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

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

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

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

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

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

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

### 3.2.3.1.3.3 Get and Set Instance-Specific Roles

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

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

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Script Task Object/Property (Read/Write Access)</th>
<th>JUEL Expression (Read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array of Strings of viewer users</td>
<td>$.roles.viewerUsers</td>
<td><code>${roles.viewerUsers}</code></td>
</tr>
<tr>
<td>Array of Strings of viewer groups</td>
<td>$.roles.viewerGroups</td>
<td><code>${roles.viewerGroups}</code></td>
</tr>
<tr>
<td>Array of Strings of context viewer users</td>
<td>$.roles.contextViewerUsers</td>
<td><code>${roles.contextViewerUsers}</code></td>
</tr>
<tr>
<td>Array of Strings of context viewer groups</td>
<td>$.roles.contextViewerGroups</td>
<td><code>${roles.contextViewerGroups}</code></td>
</tr>
<tr>
<td>Array of Strings of admin users</td>
<td>$.roles.adminUsers</td>
<td><code>${roles.adminUsers}</code></td>
</tr>
</tbody>
</table>
## Assign Roles to User

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

### Sample Code

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

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

### Sample Code

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

## Retrieve Users Assigned to a Role

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

### Sample Code

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

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

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

**Sample Code**

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

### 3.2.3.1.4 Configure Mail Tasks

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

#### Prerequisites

Configure a mail destination. See Configure the Workflow Service Mail Destination [page 19].

#### Procedure

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

   **Note**

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

4. From Mail Task Properties area, choose the for the mail task Details tab.
5. In the To field, provide a list of comma-separated mail addresses to whom you want to send the mail.

   **Note**

   For more information on character limits for SAP Cloud Platform Workflow, see Conventions, Restrictions, and Limits [page 7].

6. Optional: Add the mail addresses to the Cc for the mail and Bcc fields.
7. Choose the **Ignore Invalid Recipients** checkbox to ignore any invalid e-mail addresses maintained on mail task.

**Note**

- If you enable the checkbox, e-mail addresses that are syntactically incorrect, or that are caused by unresolvable expressions do not lead to failure of mail task, provided at least one recipient can be determined.
- If the checkbox is disabled, mail task fails when at least one invalid recipient is determined.
- If you enable the checkbox, the ignored recipients list appear in the *Monitoring Workflows* app.

8. Provide a **Subject** for the mail.

**Note**

*Subject*, *Cc*, *Bcc*, and *To* fields can contain JUEL expressions. For more information, see *Expressions* [page 77].

Except for the **Subject** field, you can either use a context reference that resolves to a string, with different mail addresses separated by commas, or a context reference that resolves to an array of mail addresses.

9. From the **Configure Mail Body** list, choose one of the following:

- **Plain Text**: Provide the message in the form of text.
- **HTML**: Create new or choose an existing HTML file with the mail content.

To create a new HTML file, perform the following steps:

1. In the **HTML Body** section, choose the **Create file** link.
2. In the **Create New File** window, provide a file name.
3. Choose **Create**.
4. In the HTML file, provide the mail content.

**Note**

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

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

5. Save and close the HTML file.

i Note

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

10. Connect the mail task to the required flow elements.

11. Choose Save.

Related Information

Conventions, Restrictions, and Limits [page 7]
3.2.3.2 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.

3.2.3.2.1 Configure Start Events

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

**Prerequisites**

You are in the process of modeling a start event. For more information, see Define Workflows [page 37].

**Context**

After the deployment of the workflow, the sample context is displayed in the Monitor Workflows app while starting a new workflow instance. For more information, see Managing Workflow Definitions [page 27].

Alternatively, you can use the API to retrieve the sample start context. For more information, see SAP Cloud Platform Workflow API.

You can propagate the user who starts a workflow instance to a service called later by a service task in the same workflow instance. For more information, see Configure Service Tasks [page 51].
Procedure

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

   Note
   ○ You can view or edit the JSON file by selecting the File link.
   ○ For more information about Code Editor, see Developing Applications.

5. Save the workflow.

Related Information

Define Workflows [page 37]

3.2.3.2.2 Configure Intermediate Message Events

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

Prerequisites

Configure a business key for your workflow. For more information about business keys, see Define Workflows [page 37].

Context

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

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

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

   **i Note**
   
   For more information on character limits for SAP Cloud Platform Workflow, see Conventions, Restrictions, and Limits [page 7].

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

   **i Note**
   
   ○ If you use a response variable, it must adhere to the syntax defined by the Java Unified Expression Language (JUEL). You can only use expressions that reference the workflow context. For more information, see Expressions [page 77].
   ○ If you don’t provide a response variable, the message is consumed by matching workflow instances. However, the context data passed by the message is not considered.

7. Choose **Save**.

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

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

**Related Information**

Conventions, Restrictions, and Limits [page 7]
3.2.3.2.3 Configure Intermediate Timer Events

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

Context

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

Procedure

1. Select Events > Intermediate Timer and drop it onto the canvas from the palette.
2. Fill in the Name and Documentation fields for the intermediate timer event.
3. In the Intermediate Timer Event Properties area, choose the Details tab.
4. Provide the waiting time interval in the Duration field.
   ○ To use expressions, choose Expression from the Duration Based On dropdown.
   
   Provide an expression in the Duration field using ISO 8601 format. For example, PT$\{context.minutes\}M. The JUEL expression $\{context.minutes\} is evaluated at runtime. You can provide multiple duration attributes by using multiple JUEL expressions. For more information about the duration formats that are supported in ISO 8601, see Conventions, Restrictions, and Limits [page 7].
   ○ To use a static value, choose Static Value from the Duration Based On dropdown. Now, provide the Duration as a numeric value, and choose a Unit of Time.
5. Choose Save.

3.2.3.3 Gateways

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

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 flow 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 72).

- **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 74).

### 3.2.3.3.1 Configure an Exclusive Gateway

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

**Procedure**

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

**Note**
A unique ID gets generated for every workflow artifact. This ID is in read-only mode.
3. On the canvas, create a sequence flow from the **Exclusive Gateway** icon to other flow objects.

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

4. Configure a condition for a sequence flow.
   For more information, see **Configure a Sequence Flow** [page 73].

5. Choose **Save**.

**Related Information**

**Accelerated Modeling with Speed Buttons** [page 76]

**3.2.3.3.1.1 Configure a Sequence Flow**

**Procedure**

1. On the canvas, choose the sequence flow you want to configure.

2. In the **Sequence Flow Properties** section provide a **Name** and **Documentation** for the flow object.

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

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

   **i Note**
   - You can mark only one outgoing sequence flow as the default.
   - If you want a certain path to execute, for example, only if an employee does not belong to Germany. You need to configure the sequence flow condition as **${context.employee.region!= "Germany"]**. For more information, see **Expressions** [page 77].

4. Choose **Save**.
3.2.3.3.2 Configure a Parallel Gateway

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

Procedure

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

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

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

Related Information

Accelerated Modeling with Speed Buttons [page 76]

3.2.4 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 75].

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

You use this procedure to deploy modeled workflows.

Prerequisites

You have the WorkflowDeveloper role. For more information, see Authorization Configuration [page 137].

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.

   i Note
   - Ensure that you save the workflow before deploying a workflow.
   - You can deploy a single workflow at a time.

3.2.6 Export Workflow Runtime Model

You can use export function to export a workflow from SAP Web IDE Full-Stack as workflow runtime model.

Procedure

1. Log in to the SAP Web IDE Full-Stack application.
2. From the left pane, choose (Development) and navigate to the Workspace folder.
3. From the context menu of your workflow file, choose Export as Workflow Runtime Model.
4. Save the exported .JSON file in your local file system.

   i Note
   Ensure that the workflow model is validated before exporting the model. If the export fails, you must check the Problems View to view the validation errors.
3.2.7 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.  
  - ![Mail Task](image): Creates a mail task. |
| ![Exclusive Gateway](image) | This speed button allows you to model the following gateways:  
  - ![Exclusive Gateway](image): Creates an exclusive gateway.  
  - ![Parallel Gateway](image): Creates a parallel gateway. |
### Speed Button Description

<table>
<thead>
<tr>
<th>Speed Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄</td>
<td>Creates a sequence flow from one flow element to another.</td>
</tr>
<tr>
<td>⏱️</td>
<td>Allows you to create a boundary timer event on a user task.</td>
</tr>
<tr>
<td>🗑️</td>
<td>Allows you to delete a flow element.</td>
</tr>
</tbody>
</table>

### i Note
When you select the sequence flow speed button, you must drag it on the element that you want to connect to. If the area is highlighted in green, then the element can be connected using a sequence flow. If the area is not highlighted, then the element cannot be connected using the sequence flow.

### 3.2.8 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). You can access data stored in the workflow context, for example, `${context.variablename}` as well as data that refers to the current task or workflow, for example, `${info.workflowInstanceId}`. The syntaxes to access this data within a JUEL expression and using the script task API are aligned. The following statements address the same attribute:

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Example of JUEL Expression</th>
<th>Example of Script Task API</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow instance context</td>
<td><code>$</code> <code>{context.employee.firstName}</code></td>
<td><code>.context.employee.firstName</code></td>
<td>[Creating and Reading Workflow Context Structures](page 58)</td>
</tr>
</tbody>
</table>
### Property Navigation and Text Expression

Property navigation and text expressions typically occur in user tasks. See [Configure User Tasks](page 39).

#### 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 73]* 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 51]*) 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 51]*) as follows:
  
  ${context.employee.empData}$

**Notices**

- When there are multiple expressions in a single field: if one of the expressions is incorrect or refers to a field that does not exist, then none of the expressions in that field are replaced. For example, in the text expression "Approval for ${context.employee.firstname} ${context.employee.lastname}". if the employee's last name field does not exist, none of the expression is replaced.

- Once expressions in texts are resolved, that is, they are replaced with the actual text at runtime, the texts are not changed if the process context changes at later point in time.

**Overview of Properties Supporting JUEL Expressions**

*Note*

All task-related properties of ${info} are only available on JUEL-enabled properties of service and user tasks.
Elements and Properties Using JUEL Expressions

<table>
<thead>
<tr>
<th>Workflow Model Element Type</th>
<th>JUEL-Enabled Property</th>
<th>Required Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence flow originating from an exclusive gateway</td>
<td>Condition</td>
<td>Boolean</td>
</tr>
<tr>
<td>Service task</td>
<td>Path</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Path to XSRF token</td>
<td>String</td>
</tr>
<tr>
<td>User task</td>
<td>Subject</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Recipient users</td>
<td>Comma-separated string or array of strings</td>
</tr>
<tr>
<td></td>
<td>Recipient groups</td>
<td>Comma-separated string or array of strings</td>
</tr>
<tr>
<td></td>
<td>HTML5 app</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Component URL</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>SAPUI5 component</td>
<td>String</td>
</tr>
<tr>
<td>Mail task</td>
<td>To</td>
<td>Comma-separated string or array of strings</td>
</tr>
<tr>
<td></td>
<td>Cc</td>
<td>Comma-separated string or array of strings</td>
</tr>
<tr>
<td></td>
<td>Bcc</td>
<td>Comma-separated string or array of strings</td>
</tr>
<tr>
<td></td>
<td>Subject</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>HTML / plain text body</td>
<td>String</td>
</tr>
<tr>
<td>Workflow model</td>
<td>Subject</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Business key</td>
<td>String</td>
</tr>
</tbody>
</table>

Handling of the Date-Related Objects

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

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

You can use the sample application to experience the workflow service offering on SAP Cloud Platform.

Context

You can create a sample application containing workflows and task user interfaces for the following scenarios:

- Employee onboarding scenario
- Document audit scenario

Procedure

1. Log in to the SAP Web IDE Full-Stack application.
2. From the left pane, choose Development (Development) and navigate to the Workspace folder.
3. Choose File ➔ New ➔ Project from Sample Application.
4. From the Sample Application Selection screen, choose Category as Workflow Sample Application.
5. Choose one of the following tiles:
   - Employee Onboarding Extension Workflow
   - Collaborative Approval Workflow Using SAP Jam
6. Choose Next.
7. Select the checkbox and provide your consent to create the application.

Results

A workflow project containing the sample application is created in your workspace.
Next Steps

Using the readme.txt file in the project, you can configure, deploy, and run the workflow sample application. For more information, see the following:

- Sample Application for Employee Onboarding Extension Workflow
- Sample Application for Collaboration in SAP Cloud Platform Workflow with SAP Jam

3.4 Creating User Interfaces

With SAP Cloud Platform Workflow you can create user interfaces for workflows.

You have the following options:

- Defining a User Interface for a Workflow [page 82]
  With SAPUI5, this option gives you a high control of the UI.
- Creating a Workflow Form [page 105]
  This option enables you to create a simple straightforward UI using predefined elements.

3.4.1 Defining 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:

- /bpmworkfloruntime
  /bpmworkfloruntime maps to the bpmworkfloruntime destination, which is configured by default for your subaccount.
  For more information, see Read Task Context Data [page 86].
  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>.
3.4.1.1 Building a Custom Task UI

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

Context

You can either define the task UI using an existing SAPUI5 component or using a form.

To build a custom task UI, execute the following steps:

Procedure

1. Creating an HTML5 Application for the Custom Task UI [page 84]
2. Deploy an HTML5 Application [page 95]

Related Information

SAP Web IDE Full-Stack Documentation
HTML5: Getting Started
Task Data in the SAP Cloud Platform, Workflow Service API Documentation
Creating a Workflow Form [page 105]
3.4.1.1 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 86]
   b. Read Task Context Data [page 86]
   c. Bind a UI Element to an Attribute of the Task Context JSON Model [page 87]
   d. Adding Task Completion Buttons [page 87]
   e. Get the Task Description [page 90]
   f. (Optional) Retrieve Parameters in Custom Task UI [page 91]
   g. (Optional) Show and Hide Footer [page 91]
   h. (Optional) Show and Hide the Back Navigation Button [page 91]
   i. (Optional) Call an External Service from a Custom Task UI [page 94]

Results

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

```xml
<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:

```javascript
init: function() {
  UIComponent.prototype.init.apply(this, arguments);
  this.setModel(models.createDeviceModel(), "device");
  var startupParameters = this.getComponentData().startupParameters;
  var taskModel = startupParameters.taskModel;
}
```javascript
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:

Sample Code

```
3.4.1.1.1 Get the Task Instance ID

Procedure

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

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

3.4.1.1.2 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.

```javascript
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);
```

3.4.1.1.3 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.
3.4.1.1.4  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>
```

3.4.1.1.5  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 87]
2. Call the Task Completion REST API [page 88]
3. Add Custom Action Buttons [page 89]
4. Update the Task List After Task Completion [page 90]

3.4.1.1.5.1  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:

```javascript
_fetchToken: function() {
  var token;
```
3.4.1.1.5.2 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
    }
  });
}
```

### 3.4.1.1.5.3 Add Custom Action Buttons

#### Procedure

1. To add buttons to the footer of the custom task UI, add these lines to the init function of `webapp/Component.js`. For example, you want to add `Approve` or `Reject` buttons:

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

2. **(Optional)** You can define the appearance of one or multiple custom action buttons as **positive** or **negative**. To do so, use the additional `type` parameter as in the following code sample:

   ```javascript
   startupParameters.inboxAPI.addAction({
     action: "REJECT",
     label: "Reject",
     type: "reject" // or "negative"// For negative appearance.
   }, function(button) {
     this._completeTask(taskId, false);
   }, this);

   startupParameters.inboxAPI.addAction({
     action: "APPROVE",
     label: "Approve",
     type: "accept" // or "positive"// For positive appearance.
   }, function(button) {
     this._completeTask(taskId, true);
   }, this);

   startupParameters.inboxAPI.addAction({
     action: "APPROVE_ALT",
     label: "Alternative Approve",
     type: "accept" // or "positive"// For positive appearance.
   }, function(button) {
     this._completeTaskAlternative(taskId, true);
   }, this);
   
   In the example above, you create one negative and two positive custom action buttons.
In case you have not specified the additional `type` parameter, the custom action button will appear as **Default**.

### 3.4.1.1.5.4 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
Sample Code

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

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

#### 3.4.1.1.6 Get the Task Description

To retrieve the task description:

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

```xml
Sample Code

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

The `init` function of webapp/Component.js should be like this:

```javascript
Sample Code

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

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

    var contextModel = new sap.ui.model.json.JSONModel("/
bpmworkflowruntime/rest/v1/task-instances/" + taskId + "/context");
    contextModel.setDefaultBindingMode(sap.ui.model.BindingMode.OneWay);
```
3.4.1.1.1.7 Retrieve Parameters in Custom Task UI

To retrieve the parameters, add the following lines:

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

3.4.1.1.1.8 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);
},
```

3.4.1.1.1.9 Show and Hide 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);
},
```
var startupParameters = this.getComponentData().startupParameters;
startupParameters.inboxAPI.setShowNavButton(true);
},

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);
},
```

### 3.4.1.1.1.10 Disable and Enable Custom Action Buttons

**Prerequisites**

- You have implemented a custom task UI and the target SAPUI5 application is embedded into the detail view of *My Inbox*.
- You have added custom action buttons via `addAction` method available in the My Inbox API. For more information, see *Add Custom Action Buttons* [page 89]
**Context**

For example, you have implemented validation logic for user input data. Based on the validation result, you may wish to disable or enable the action buttons, available in the *My Inbox* button bar, added in the prerequisites.

**Procedure**

1. To disable a single custom action button available in the *My Inbox* button bar, you can use the `disableAction()` method of *My Inbox API*. The method `disableAction()` expects as parameter the value of the `action` parameter used in the `addAction()` method. For example, if you have added a button via `addAction({action: "APPROVE", label: "Approve"})`, you can disable it via `disableAction("APPROVE")`. In this case the implementation for your event handler should look like this:

```javascript
onEventHandler: function(){
    var startupParameters = this.getComponentData().startupParameters;
    startupParameters.inboxAPI.disableAction("APPROVE");
},
```

2. To enable a single custom action button available in the *My Inbox* button bar, you can use the `enableAction()` method of *My Inbox API*. The method `enableAction()` expects as parameter the value of the `action` parameter used in the `addAction()` method. For example, if you have added a button via `addAction({action: "APPROVE", label: "Approve"})`, you can enable it via `enableAction("APPROVE")`. In this case the implementation for your event handler should look like this:

```javascript
onEventHandler: function(){
    var startupParameters = this.getComponentData().startupParameters;
    startupParameters.inboxAPI.enableAction("APPROVE");
},
```

3. To disable all custom action buttons available in the *My Inbox* button bar, the implementation for your event handler should be the following:

```javascript
onEventHandler: function(){
    var startupParameters = this.getComponentData().startupParameters;
    startupParameters.inboxAPI.disableAllActions();
},
```

4. To enable all custom action buttons available in the *My Inbox* button bar, the implementation for your event handler should be the following:

```javascript
onEventHandler: function(){
```
Results

The buttons available in the My Inbox button bar become disabled or enabled as per your implementation.

3.4.1.1.11 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. This route targets a destination pointing to your service, for example, deployed as a Java application on SAP Cloud Platform.

   `Sample Code`

   ```json
   ...
   {
     "path": "/external-service",
     "target": {
       "type": "destination",
       "name": "external-service",
       "entryPath": "/",
     },
     "description": "External Service"
   }
   ...
   ```

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.
### i 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: ""
    });
}
```

### 3.4.1.2 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.

### 3.4.1.2 Creating a 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 96]
2. Define the Destination Route [page 96]
3. Extend the View [page 97]
4. Extend the Controller [page 98]

Related Information

SAP Web IDE Full-Stack Documentation
HTML5: Getting Started

3.4.1.2.1 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.

3.4.1.2.2 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:

```
{   "path": "bpmworkflowruntime",
```
3.4.1.2.3   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:

```xml
<PAGE title="Workflow Start UI">
  <CONTENT>
    <VBOX width="100%" direction="Column" id="__vbox0">
      <ITEMS>
        <INPUT width="100%" id="textInput" value="/{text}" />
        <BUTTON text="Start Workflow" width="100px" id="__button0" press="startWorkflow" />
        <TEXT text="/{result}" maxLines="0" id="__text0" />
      </ITEMS>
    </VBOX>
  </CONTENT>
</PAGE>
```
3.4.1.2.4 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 99]
- Implement XSRF Token Fetch [page 99]
- Implement Workflow Instance Start [page 100]
- Bind an Action to the Button Push Event [page 101]

Results

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

```
{ 
  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
      },
      data: JSON.stringify({
        definitionId: "<your workflow ID>",
        context: {
          text: text
        }
      }),
      success: function(result, xhr, data) {
        model.setProperty("/result", JSON.stringify(result, null, 4));
      }
    });
  }
}
```
3.4.1.2.4.1 Implement Data Model Instantiation

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

```
Sample Code

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

3.4.1.2.4.2 Implement XSRF Token Fetch

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

```
Sample Code

_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;
}
```
3.4.1.2.4.3 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
_startInstance: function(token) {
    var model = this.getView().getModel();
    var inputValue = model.getProperty("/text");
    $.ajax({
        url: "/bpm/workflowruntime/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));
        }
    });
}
```

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

Feel free to change the name of the text field of the workflow context to fit to the corresponding workflow definition.
3.4.1.2.4.4 Bind an Action to the Button Push Event

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

**Sample Code**

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

3.4.1.3 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 102].</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>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>inboxAPI</td>
<td></td>
<td>Contains the API published by My Inbox for integration purposes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, see <a href="#">My Inbox UI Integration API Reference</a> [page 103].</td>
</tr>
</tbody>
</table>

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

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

### 3.4.1.3.1 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.
3.4.1.3.2 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.

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.

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.

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.

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 Instanceld 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.

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.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showNavButton</td>
<td>boolean</td>
<td>Flag representing whether to show or hide navigation button in header of the page. The default value is false.</td>
</tr>
<tr>
<td>navEventHandler?</td>
<td>function</td>
<td>Function to be called when navigation button is clicked.</td>
</tr>
</tbody>
</table>

### 3.4.2 Creating a Workflow Form

With the workflow form editor you can create UIs for your end user.

A form consists of the following:
- form header
- form details

The form header is filled with information from the user task’s runtime attributes defined in the form editor, for example, *Created On* and *Created By*. In addition, there is the information defined in the workflow editor, for example, *Name*, *Subject*, and *Description*.

Whereas the Form Details section is filled using your definition in the form editor. You can model fields and also define a layout by grouping the fields into sections and subsections.

### 3.4.2.1 Create Your Form

You can create forms using the form editor.

### Context

Either you access the form editor directly as described in the procedure below or you access it from within the workflow editor. For more information on using the workflow editor, see [Configure a Task User Interface Using Workflow Forms](page_47).

**i Note**

To work comfortably with the fields or decisions table, you can use the slider below each table to adapt the table height.
Procedure

1. (Optional) Create a folder in your workflow project for the forms of your workflow project and name it, for example, forms.
   You could use any folder.
2. Right-click the workflow project or any folder within your project, for example, a dedicated forms folder, and choose New Form.
3. Enter a name, ID, and a revision for your form, for example:

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ApprovalForm</td>
</tr>
<tr>
<td>ID</td>
<td>approval-form</td>
</tr>
<tr>
<td>Revision</td>
<td>2.0 or Draft</td>
</tr>
</tbody>
</table>

4. Choose Create.

Results

A corresponding file with the name <yourformname>.form is created, and the form editor opens an empty form.

To reopen a form file, either double-click it or choose Open With Form Editor.

You can rename the file at any time.

i Note

- If you have already referenced the form file within a user task of a workflow, make sure to adapt the reference in the workflow editor accordingly.
- The form ID must be unique inside your account. Do not change the form ID unless you are sure that you want to give the form a new identity.
3.4.2.1.1 Define Forms by Adding Fields

You build forms using fields that you can arrange using sections and subsections.

Procedure

1. To open your form file, double-click it.
2. To add a field to your form, choose Add Field at the top of the field table.
   
   If an existing field is selected, the new field is inserted right after the selected field. If no field is selected, the new field is added depending on which element is selected. If a section is selected, the new field is added at the end of the section. If a subsection is selected, the new field is added to the end of the subsection. For more information, see Adapt the Form Layout [page 111].
3. Enter the text for the field label.
4. Enter the ID of the field or use the automatically generated one.

   **Note**
   
   IDs must start with a letter and must contain only alphanumeric characters and underscores. IDs must be unique within a section or subsection. If a form does not contain sections or subsections, the IDs must be unique within the whole form.

5. Bind your field to a property element of the task context model.

   When you bind a field to a property in the task context, the respective value is shown during form rendering. Furthermore, if the field is set to editable (see Set the mode of your field [page 109]), changes to that value by the user are written back to the task context during task completion. When the bound property does not exist in the task context, it might be created during task completion. For more information and limitations, see Automatic Model Initialization and Model Cleanup [page 113].

   The syntax follows the JUEL style described in Expressions [page 77].

   **Note**
   
   You can only access the workflow instance context using dot notation. Conditions and literals are not supported. Make sure to use a valid path to a property in the context.

   **Example**
   
   Let’s assume that your process context is the following:

   ```json
   { 
     "report": { 
       "name": "Travel for TechEd Las Vegas", 
       "id": "A2E6D6A5ABD4C37", 
       "owner": "Steve Consultant", 
       "totalClaimedAmount": 870.30, 
   ```
You then want to define a field that displays the timestamp of the purchase order in the invoice. You can use the following data for this field:

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label/Title</td>
<td>Date and Time of Purchase Order</td>
<td>-</td>
</tr>
<tr>
<td>Type</td>
<td>DateTime</td>
<td>-</td>
</tr>
<tr>
<td>Value</td>
<td>${\text{context.report.invoices[0].orderDatetime}}$</td>
<td>Here, &quot;value&quot; is short for &quot;value binding&quot;.</td>
</tr>
</tbody>
</table>

6. Set the type for your field.

Field types determine how the field is represented in your task UI. Furthermore, the task UI validates the user input against the value range of the specified type.

Currently the following field types are supported:

<table>
<thead>
<tr>
<th>Type</th>
<th>Value Range</th>
<th>UI Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Any printable character</td>
<td>Labeled input field</td>
</tr>
<tr>
<td>Boolean</td>
<td>false or true</td>
<td>Checkbox</td>
</tr>
<tr>
<td>Integer</td>
<td>-9007199254740991 to 9007199254740991</td>
<td>Labeled input field</td>
</tr>
<tr>
<td>Float</td>
<td>-3.4028235e+38 to 3.4028235e+38</td>
<td>Labeled input field</td>
</tr>
<tr>
<td>Date</td>
<td>Any date of Gregorian calendar from year 1 to year 9999</td>
<td>Labeled input field with date picker</td>
</tr>
<tr>
<td>Datetime</td>
<td>Any point in time within a date</td>
<td>Labeled input field with date and time selector</td>
</tr>
<tr>
<td>Type</td>
<td>Value Range</td>
<td>UI Representation</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>Time</td>
<td>Any time of a day, from 00:00:00 to 23:59:59 (or 12:00:00AM-11:59:59PM depending on the locale)</td>
<td>Labeled input field with time selector</td>
</tr>
</tbody>
</table>

7. Set the height for your field.

Currently the following field heights are supported:

<table>
<thead>
<tr>
<th>Height</th>
<th>UI Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Line (Default)</td>
<td>A single line input field</td>
</tr>
<tr>
<td>Small</td>
<td>A text area around twice the height of a single line field with scrolling capabilities</td>
</tr>
<tr>
<td>Medium</td>
<td>A text area around twice the height of a small field with scrolling capabilities</td>
</tr>
<tr>
<td>Large</td>
<td>A text area around twice the height of a medium field with scrolling capabilities</td>
</tr>
</tbody>
</table>

8. Set the mode of your field.

Currently the following modes are supported:

<table>
<thead>
<tr>
<th>Mode</th>
<th>UI Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editable (Default)</td>
<td>The end user can modify the value on the UI.</td>
</tr>
<tr>
<td>Display-Only</td>
<td>The end user can no longer modify the value on the UI. The field is in display mode.</td>
</tr>
</tbody>
</table>

**Note**
The mode only affects the rendered form and not the workflow runtime itself. Read-only attribute values can still be modified, for example, using the REST API or script tasks.

If the complete form is set to read-only mode, it cannot be changed (see Create Your Form [page 105]).

9. (Optional) Add field placeholders.

For an editable field, you can define a placeholder. When the control has no value, the placeholder gives users a hint when they enter data.

10. Set the constraints of your field.

Currently the following constraint is supported for editable fields:

<table>
<thead>
<tr>
<th>Constraint</th>
<th>UI Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>The field is marked as required. The end user must enter a value. Otherwise, the end user cannot use decisions to complete the task (see Add or Delete Decisions [page 112]).</td>
</tr>
</tbody>
</table>
### 3.4.2.1.2 Set the Form Mode

After you created your form, you can choose whether end users are allowed to change the values in the form's fields.

**Procedure**

1. In the fields table, make sure that no field is selected.

   - **i Note**
     
     To deselect a field, click it again.

2. Set the mode of your form.

   Currently the following modes are supported:

<table>
<thead>
<tr>
<th>Mode</th>
<th>UI Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editable (Default)</td>
<td>For each field, you can modify the value on the UI</td>
</tr>
</tbody>
</table>
   | Display-Only             | For each field, you can no longer modify the value on the UI. The fields are in display mode.

A form wide read-only mode overwrites individual field modes with read-only mode. You can no longer change the mode for individual fields. However, your previously modeled modes as well as constraints on the individual fields are preserved in case you switch to "Edit" mode again.

- **i Note**
  
  The mode only affects the rendered form and not the workflow runtime itself. You can still modify read-only attribute values on API level or in script tasks.
3.4.2.1.3 Adapt the Form Layout

You must define the layout of your forms, for example, whether to group fields.

Group Fields into Sections and Subsections

To group your fields, choose Add Section. If your form does not contain any section yet, this action moves all fields into a new section. Otherwise, a new section is added.

If you need a more fine granular grouping, you can divide your sections into subsections by choosing Add Subsection while a section is selected. If the selected section already contains fields, they are moved into the new subsection.

i Note

Do not add more than 100 sections to your form and not more than 100 subsections to a single section. If you need more than 100 subsections, divide them up across several sections.

If you need more than 100 sections, split your UI into multiple forms connected via multiple human tasks.

Also do not use more that 100 fields per form (with or without sections and subsections). Only 100 fields per section or subsection are supported.

To add new fields to a section or subsection, select it and choose Add Field.

For each section or subsection, you can specify a text that is used as a section title.

Move Form Elements

To change the location of fields, sections, and subsections, you have the following options:

- Use the context menu
- Use the actions in the table toolbar (copy, cut, paste)
- Use drag and drop

If you paste a subsection in a place where only sections are supported, it is converted into a section and the other way around. The latter one only applies to sections containing fields only.
3.4.2.1.4 Add or Delete Decisions

You must define what decisions users can choose from to complete a task.

Context

Users can complete tasks using decision buttons. You can model the following types of decisions for your form:

- A positive decision
  Example: Approve
- A negative decision
  Example: Reject
- A neutral decision

Every type has its own visual appearance on your form matching its semantics.

The workflow context stores what decision the user has selected. For more information about how to access the decision, see Access the Decisions [page 113].

Procedure

1. Switch to Decisions.
2. Choose Add.
   a. Specify the display text of the decision.
   b. Specify the ID of the decision.
   c. Specify the type of the decision.

Move Decisions

Procedure

To duplicate or change the order of decisions, you have the following options:

- Use the context menu
- Use the actions in the table toolbar (copy, cut, paste)
- Use drag and drop

Note

MyInbox sorts the decisions by type (Positive, Negative, Neutral) first and only then takes into account what position they have in the table.
3.4.2.1.5 Access the Decisions

The business user chooses a decision button to complete a user task.

Context

For user tasks using forms, the decision is stored within the user task’s properties. For more information, see Expressions [page 77]. For each task in the flow that was completed the decision_id of the last decision of each task is stored in the decision property.

Example

Assuming the end user choses Accept in a user task with the ID "usertask1". Then you can access the corresponding decision ID, for example, accept using a JUEL expression, as follows:

```plaintext
${usertasks.usertask1.last.decision}
```

To use a decision in the context of a gateway, proceed as follows:

Procedure

1. Select the sequence flow that should have a condition that is linked to a decision.
2. Enter your expression like this: "${usertasks.usertask1.last.decision=="accept"}".

3.4.2.2 Automatic Model Initialization and Model Cleanup

This section describes the runtime behavior of forms.

Model Initialization

To render a form in the UI, the corresponding data model must be built in such a way that SAP UI5 can render it. Therefore, the following is checked:

- Binding Collisions
  When a binding collision occurs, the UI does not render, but an error message is displayed. In addition, the workflow forms runtime posts an aggregated issue report to the browser console.
- Referenced objects in binding path are missing
You can bind fields to context properties that do not exist in the workflow task context when the form is opened. Then, the workflow forms runtime creates the missing context properties.

**i Note**

For binding paths including collection elements, missing context properties can only be created if each collection element within the path exists in the context.

For more information, see [Bind your field to an attribute of the task context model](#). This is what it looks like:

<table>
<thead>
<tr>
<th>Workflow (Task) Context</th>
<th>Binding</th>
<th>UI Model (Initial)</th>
<th>UI Model with Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>${context.myNode1.myNode2.myProperty}</td>
<td>{ myNode1: {} }</td>
<td>{ myNode1: {    myProperty: &quot;anyValue&quot;    } }</td>
</tr>
<tr>
<td>a: [{}]</td>
<td>${context.a[0].b.c}</td>
<td>anyValue</td>
<td>{ a: [{ b: { c: &quot;anyValue&quot; } } ] }</td>
</tr>
</tbody>
</table>

This also applies for collection elements.

As an example, the following scenario is not supported, because the specified collection element (0) is not existing:

<table>
<thead>
<tr>
<th>Workflow (Task) Context</th>
<th>Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>a: []</td>
<td>${context.a[0].b.c}</td>
</tr>
</tbody>
</table>

**Model Cleanup**

When the form is completed, the workflow form runtime performs a model clean-up before storing data to the workflow. The clean-up considers the following:

- No unnecessary information is stored in the workflow context.
When you access the information in subsequent JUEL expressions or JavaScript code, you don't have to manage nonexisting information. That is, in a JUEL expression you can check `{context.myObject.myProperty} == 'myValue'`. This is the same behavior as in a JavaScript statement.

This is how the workflow forms runtime handles the different data types:

### 3.4.2.2.1 String

<table>
<thead>
<tr>
<th>Workflow(Task) Context Before</th>
<th>Binding</th>
<th>User Operation</th>
<th>Workflow Context After</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>$</td>
<td>None</td>
<td>{}</td>
</tr>
<tr>
<td></td>
<td>{context.myProperty}</td>
<td>Enter value</td>
<td>{}</td>
</tr>
<tr>
<td></td>
<td>{&quot;myObject&quot;:{&quot;myProperty&quot;:&quot;&lt;uservalue&gt;&quot;}}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{&quot;myObject&quot;:{&quot;myProperty&quot;:&quot;&quot;}}</td>
<td>Enter value</td>
<td>{}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{&quot;myObject&quot;:{&quot;myProperty&quot;:&quot;myValue&quot;}}</td>
<td>Field changed</td>
<td>{}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{&quot;myObject&quot;:{&quot;myProperty&quot;:&quot;&quot;}}</td>
<td>Field cleared</td>
<td>{}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.4.2.3 Version Forms

Versioning is a key activity any developer deals with when building production-grade software.

This holds specifically true for forms used by potentially long-running workflows. Without versioning, changes you develop for forms in future workflow instances unexpectedly also affect already running instances. These unintended changes often have a negative impact.

Comparison to Versioning with Git or Similar Tools

Versioning in this context is not to be confused with versioning using version control systems (VCS). Versioning forms in a Git repository deals with design-time versioning of artifacts and is orthogonal to the runtime-related versioning discussed here. See below for recommendations on how to combine the two.

Technical Versioning of Forms

By default, forms in and of themselves are already versioned in a technical way: Every time a form is deployed to the form runtime, a new (technical) version is created for it. Previous versions are preserved for historical
and auditing reasons but end users cannot access them at runtime. This way, developers and administrators have transparency over who deployed which form and when.

**Compatible Changes Compared to Incompatible Changes**

With the technical versioning outlined above, any change to a form represents a new (unqualified) version. Thus, the developer cannot distinguish between compatible and incompatible changes.

If it fixes a usability or functional issue, a change or release is typically considered compatible. Whereas incompatible changes fundamentally alter a form and are usually driven by a business requirement. This could, for example, apply to mandatory form fields that you add to a form. Consequently, a workflow needs to store additional data in its context that are expected by the changed form. To address this, you as the developer typically change the workflow definition accordingly, too. These incompatible changes then are assumed to take effect for new workflow instances while already running workflow instances continue to operate on the previous version. Already running workflow instances would usually not have the necessary context data.

**Forms Revision Concept**

To allow the differentiation between compatible and incompatible changes, each form has a revision property that is stored along with any other properties, for example, the form name and form ID. On deployment of a form, it signifies the major release to which the newly created version is assigned to. Conversely, simply changing the revision without deploying the form has no effect.

You can set the revision while creating a new form or when editing its metadata. For more information, see Create Your Form [page 105].

When you refer to a form in a workflow’s user task, you are asked to specify the revision of the form to use. For more information, see Configure a Task User Interface Using Workflow Forms [page 47].

As stated above, changing the revision of a form and deploying it to the form runtime implies a major release of the form. By contrast, deploying a form without a change of its revision implies a minor release. This way, you can choose between changes that affect existing workflow instances and changes that only affect future workflow instances. Provided that you change the revision of the respective workflows’ user tasks accordingly.

**Versioning Best Practices**

The following is a list of recommendations of when to patch a form (leaving the revision unchanged) and when to create a new revision (altering the revision).

<table>
<thead>
<tr>
<th>Compatible Changes (Revision Unchanged)</th>
<th>Incompatible Changes (Revision Altered)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the label or placeholder of a form field</td>
<td>Change the type of a form field</td>
</tr>
</tbody>
</table>
Compatible Changes (Revision Unchanged) | Incompatible Changes (Revision Altered)
--- | ---
Change the layout settings of a form, for example, sections or sub-sections | Change the ID or value of a form field (*)
Change the text or type of a form field | Change the decision ID of a form field
Remove a read-only form field | Add or remove a form field (*)

(*) There might be conditions where compatibility can be ensured. But this is usually an incompatible change (to be decided case-by-case).

→ Tip
When changing the revision of a form, we recommend that you tag or otherwise flag the corresponding commits in Git. This helps when you need to patch an older revision later on.

Related Information
Define Forms by Adding Fields [page 107]

3.4.2.4 Deploy Your Form

As with workflows, you must deploy a form to the form runtime before you can use it. Contrarily to workflows, you must deploy a form with a revision.

Prerequisites

- You have the WorkflowDeveloper role. For more information, see Authorization Configuration [page 137].
- All property values you set are valid. Otherwise, the deployment fails.
- You maintained all mandatory properties for fields, sections, and subsections. Otherwise, the deployment fails.

Procedure

Right-click the forms file, and choose Deploy > Deploy to SAP Cloud Platform Workflow.
3.5 Using Workflow APIs

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

SAP Cloud Platform Workflow exposes two kinds of API to address different use cases. The OData-based API exposes user-task related data implementing a subset of the Task Consumption Model (TCM), see SAP Note 2304317. Its primary use case is to build-up a personal inbox. The REST-based API allows you to list and manage workflow instances, definitions, and user tasks across recipients. Depending on your role, you can execute 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.
- Execute various lifecycle and administrative operations on the resources involved in the workflow service.

For more information about who can execute these actions, see Authorization Configuration [page 137].

For more information about using these APIs, see the Workflow Service REST API documentation and Workflow Service Odata 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 Bearer Assertion Flow for OAuth 2.0)

To ensure optimal operation of the service, REST API execution is subject to resource limits, for example, regarding the number of requests per second. If the limits are exceeded, API calls return HTTP status 429 ("Too many requests"). The client should then reduce the number of calls.

Related Information

Managing Workflows Using the Monitor Workflows App [page 23]
Access Workflow APIs Using OAuth 2.0 Authentication (Client Credentials) [page 120]
3.5.1 Access Workflow APIs Using OAuth 2.0 Authentication (Client Credentials)

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

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. Register an OAuth client.
   a. Navigate to your subaccount in the SAP Cloud Platform cockpit.
      For more information, see [Navigate to Global Accounts and Subaccounts in the Cockpit](#).
   b. In the navigation area, choose [Security > OAuth](#).
   c. Under [OAuth Settings](#), choose [Clients](#).
   d. To create a client, choose [Register New Client](#) and use the following data, then choose [Save](#).

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Free text</td>
<td></td>
</tr>
<tr>
<td>Subscription</td>
<td>&lt;SAP provider account&gt;/bpmworkfloruntime</td>
<td>Creates the OAuth 2.0 client in the context of your workflow service subscription.</td>
</tr>
<tr>
<td>ID</td>
<td></td>
<td>This is your client ID used for authorization.</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>
<tr>
<td>Secret</td>
<td>Free text</td>
<td>This is your client secret used to access APIs.</td>
</tr>
<tr>
<td>Token Lifetime</td>
<td>Default: 60 minutes</td>
<td>You can adapt the value.</td>
</tr>
</tbody>
</table>

For more information, see [Register an OAuth Client](#).
2. To call the user `oauth_client_<clientID>`, assign the necessary role of the workflow service API.
   a. Navigate to your subaccount in the SAP Cloud Platform cockpit.
      For more information, see Navigate to Global Accounts and Subaccounts in the Cockpit.
   b. In the navigation area, choose Services.
   c. Search for the Workflow service.
   d. On the Workflow tile, choose Configure Service.
   e. In the navigation area, choose Roles.
   f. In the Roles table, select the role that you want to assign to the `oauth_client_<clientID>` user,
      where clientID is the ID of the OAuth client that you have just created.
      For more information about roles, see Authorization Configuration [page 137].

3. 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.

4. Perform the call to the workflow service API by sending the access token as the header:
   ○ Header name: Authorization
   ○ Header value: Bearer <access token>
   The only difference from basic authorization is the header where you use the authorization header as:
Bearer <access-token>. The rest all remains the same.

**Note**

Consume the API using the OAuth 2.0 authorization protocol.

There is no need for an XSRF token if you are using the OAuth 2.0 authentication protocol. All you need
is to get the access token first:
   ○ URL: Token endpoint URL
   ○ Method: POST
   ○ User name: OAuth client ID
   ○ Password: OAuth client secret
   The access token that you receive is used to call the APIs.
3.5.2 Access Workflow APIs Using OAuth 2.0 Authentication (Authorization Grant)

This procedure illustrates how to call workflow service APIs using OAuth 2.0 authentication using an example walk-through of the authorization code flow. It shows how several OAuth2 concepts are specifically applied to workflow service and which configuration parameters are used.

Prerequisites

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

<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 137].

Context

Developers typically use this flow in web applications. However, other flows might be supported or more appropriate in your use case. See, for example, a blog about another flow.

Procedure

1. Request an authorization code from the OAuth 2.0 authorization server.
   
   a. Send a GET request to the authorization endpoint and specify both the client ID and the response type as "code". To determine the endpoint URL in the cockpit, see Security ➔ OAuth ➔ Branding ➔ OAuth URLs.

   Example: https://oauthasservices-<your_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 response redirects to the URL that you specified as a callback URL in the client details. The value of the parameter code represents the access token.

c. Copy the code from the HTTP URL.

2. Request an access token from the OAuth 2.0 authorization server.

a. Send a POST request to the token endpoint and specify the grant type as authorization code. To determine the endpoint URL in the cockpit, see Security > OAuth > Branding > OAuth URLs. Use the url service configuration parameter and the code from step 1c.

Example: https://oauthasservices-<your_account>.<landscape_host>/oauth2/api/v1/token?grant_type=authorization_code&code=<code_step1c>

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

c. Copy the access token from the HTTP response body (access_token attribute of the JSON structure).

\[\text{i Note} \]

If the access token expires before you get to execute step 3, use a refresh token.

The HTTP response in step 2 includes a refresh token (refresh_token attribute). It is typically used when the lifetime of the returned access token has expired but the application still wants to execute an HTTP request (as in step 3) on behalf of the given user. You can use the refresh token to request a new access token for the user without again asking the user for consent. The new access token then replaces the old one with a new lifetime.

To request a new access token for a given refresh token, send a POST request to the same token endpoint as in step 2 passing the refresh token. The call must be authenticated again with basic authentication, where the user name corresponds to your OAuth client ID and the password to the client secret.

\[\text{Example} \]

https://oauthasservices-<your_account>.<landscape_host>/oauth2/api/v1/token?grant_type=refresh_token&refresh_token=<refresh_token>

However, it is important to understand that the refresh token has a lifetime as well. Lifetimes of access and refresh tokens can be configured separately. If the lifetime of the refresh token has expired, there is no means to request a new refresh token.

3. Perform the call to the workflow service API by sending the access token as the header. Use the end-points below the base URL from the service configuration parameter workflow_api_url.

- Header name: Authorization
- Header value: Bearer <access token>
3.5.3 Determine the Service Host

In the Neo environment, 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.

**i Note**

If you access the workflow APIs from a user interface of an application, you typically need to use a URL that enables Cross-Origin Resource Sharing (CORS) through reverse proxies.

- In the Neo environment:
  - If you develop a custom task UI using an HTML5 application, 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 84].
  - 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 96].

**Related Information**

Determine the Service Host in Neo Environment [page 124]

3.5.3.1 Determine the Service Host in Neo Environment

**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.

3.5.4 Modifying the Context of a Workflow Instance

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

**i Note**

- If the context of a workflow instance is in COMPLETED or CANCELED status, the system does not allow you to modify it.
• We recommend suspending the workflow instance first and ensure that further entries are not written into the corresponding execution log. Then the context modification is considered safe from collisions with any ongoing workflow instance activities. After the necessary changes to the context are performed, you can resume the workflow instance execution. See the section about suspending or resuming workflow instance. For more information, see /v1/workflow-instances/{workflowInstanceId}.

Override Context

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

Example

Context contents before overriding:

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

Override operation payload:

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

Context contents after the override operation (equals the payload of the override operation):

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

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

The following situations are possible in this case:

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

ℹ️ Note
Merging happens at all levels of complex objects nesting.

💡 Example

Context contents before patching:

```javascript
{  
  variableOnlyInOldContext: 1,
  variableOverriden: "good bye!",
  variableNestedObject: {
    variableNested: true,
    variableNestedInOldContext: 1000
  },
  variableNew: "I'm new"
}
```

Patch operation payload:

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

Context contents after the override operation:

```javascript
{  
  variableOnlyInOldContext: 1,
  variableOverriden: "hello!",
  variableNestedObject: {
    variableNested: false,
    variableNestedNew: "new value"
  },
  variableNew: "I'm new"
}
```
Consider the naming conventions for context variables. For more information, see Conventions, Restrictions, and Limits [page 7].

### 3.5.5 Updating Task Properties

With the **task patch API**, you can modify the properties of the tasks in status READY or RESERVED.

To update a task, 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>",
    "dueDate": "<New due date>",
    "priority": "<New priority>",
    "processor": "<New processor>",
    "recipientUsers": "<New recipient users>",
    "recipientGroups": "<New recipient groups>"
}
```

Where `<New subject>`, `<New description>`, `<New due date>`, `<New priority>`, `<New processor>`, `<New recipient users>`, and `<New recipient groups>` refer to the values of the task subject, description, due date, priority, processor, recipient users, and recipient groups after the operation is performed.

Although this sample includes all fields, you only need to specify those fields that you really want to change.

For the "priority" field, the following values are supported:

- "LOW"
- "MEDIUM"
- "HIGH"
- "VERY_HIGH"

You can specify the due date using either of these formats: `yyyy-MM-dd'T'HH:mm:ss[.SSS] 'Z'` or `yyyyMMddHHmmss[.SSS]`. The specified time stamp is UTC and is shown to the users in My Inbox in their local time.
Supported date values are, for example:

- 2018-02-17T12:28:51Z
- 20180217122851
- 20180217122851.854

**Note**
The workflow service does not explicitly check whether processors, recipient users, or groups assigned to user tasks actually exist in the system.

**Sample Code**

```json
{
    "subject": "Approve purchase of the new monitor for John Doe",
    "description": "John Doe has requested a new monitor, because the old one has been broken",
    "priority": "MEDIUM",
    "dueDate": 20180217122851,
    "processor": "JaneDoe",
    "recipientUsers": "JaneDoe, AlexSmith",
    "recipientGroups": "Managers, HRs"
}
```

To remove a due date, recipient users, recipient groups, or the processor from a task, use an empty string:

**Sample Code**

```json
{
    "dueDate": "",
    "processor": "",
    "recipientUsers": "",
    "recipientGroups": ""
}
```

**Expressions**

You can use [Expressions](page 77) to refer to the context of the relevant workflow instance while updating the task properties:

**Example**

Task Update Payload with Expressions

**Sample Code**

```json
{
    "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:

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

The task has the subject **Approve purchase order for John Doe** 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**. The **Workflow Participant** role must be assigned to your user. To this end, "status" ("COMPLETED") and optionally "context" need to be present in the payload, for example:

```json
{
  "context": {
    "price": 6000,
    "reductionReason": "Outdated"
  },
  "status": "COMPLETED"
}
```

To update and complete the task with the same request, the **Workflow Administrator** role must be assigned to your user. The payload then looks as follows:

```json
{
  "context": {
    "price": 6000,
    "reductionReason": "Outdated"
  },
  "status": "COMPLETED",
  "subject": "Approve purchase order for $\{context.employee.name\} $\{context.employee.surname\} $\{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 Doe", but the description is set taking into account the new values: "Price: 3000 EUR".

**Related Information**

Authorization Configuration [page 137]

### 3.5.6 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.

**Logged Entries/Events**

<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>WORKFLOW_CONTEXT_OVERWRITTEN_BY_ADMIN</td>
<td>Context administrator completely overrode the workflow context.</td>
</tr>
<tr>
<td>WORKFLOW_ROLES_PATCHED_BY_ADMIN</td>
<td>Administrator changed the instance-specific role assignment of the given workflow instance.</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>
</tbody>
</table>
### Log Entry/Event

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERTASK_PATCHED_BY_ADMIN</td>
<td>User task status, its properties, or its context was changed by administrator.</td>
</tr>
<tr>
<td>USERTASK_CANCELED_BY_BOUNDERY_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>MAILTASK_CREATED</td>
<td>Mail task created.</td>
</tr>
<tr>
<td>MAILTASK_COMPLETED</td>
<td>Mail task completed.</td>
</tr>
<tr>
<td>MAILTASK_FAILED</td>
<td>Mail 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>

### 3.5.7 Error Codes

If an error occurs while working with the SAP Cloud Platform Workflow API, the returned error object has an "errorCode" attribute.

This attribute identifies the area or workflow element where the problem occurred.

The table below describes the error code groups and points to the documentation that helps you fix the error.
<table>
<thead>
<tr>
<th>Error Code / Error Code Prefix</th>
<th>Description</th>
<th>Related Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>bpm.workflowruntime.generic.error</td>
<td>This is a generic error. Contact SAP support citing the given log ID.</td>
<td>Troubleshooting [page 147]</td>
</tr>
<tr>
<td>bpm.workflowruntime.expression</td>
<td>There was a problem resolving an expression. This might be caused by the</td>
<td>Expressions [page 77]</td>
</tr>
<tr>
<td></td>
<td>expression in the workflow definition or the data accessed through an</td>
<td></td>
</tr>
<tr>
<td></td>
<td>expression, for example, the workflow context.</td>
<td></td>
</tr>
<tr>
<td>bpm.workflowruntime.destination</td>
<td>There was a problem resolving or accessing a destination.</td>
<td>Destinations [page 139]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configure the Workflow Service</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mail Destination [page 19]</td>
</tr>
<tr>
<td>bpm.workflowruntime.service.task</td>
<td>There was a problem executing a service task.</td>
<td>Configure Service Tasks [page 51]</td>
</tr>
<tr>
<td>bpm.workflowruntime.mailtask</td>
<td>There was a problem executing a mail task.</td>
<td>Configure Mail Tasks [page 65]</td>
</tr>
<tr>
<td>bpm.workflowruntime.mailtask.connection</td>
<td>There was a problem connecting to the mail server, either on network level</td>
<td>Configure Mail Tasks [page 65]</td>
</tr>
<tr>
<td></td>
<td>or relating to the secure communication setup.</td>
<td></td>
</tr>
<tr>
<td>bpm.workflowruntime.mailtask.server</td>
<td>There was a problem while communicating with a mail server. The server</td>
<td>Configure Mail Tasks [page 65]</td>
</tr>
<tr>
<td></td>
<td>refused the login or did not accept a mail.</td>
<td></td>
</tr>
<tr>
<td>bpm.workflowruntime.scripttask</td>
<td>There was a problem executing a script task.</td>
<td>Configure Script Tasks [page 56]</td>
</tr>
<tr>
<td>bpm.workflowruntime.usertask</td>
<td>There was a problem executing a user task.</td>
<td>Configure User Tasks [page 39]</td>
</tr>
<tr>
<td>bpm.workflowruntime.rest</td>
<td>There was a problem calling the REST API.</td>
<td>Using Workflow APIs [page 119]</td>
</tr>
</tbody>
</table>

**Related Information**

API Hub: SAP Cloud Platform Workflow

---

"SAP Cloud Platform Workflow Development"
4  User Guide

Introduction [page 133]
Working with Tasks in My Inbox [page 133]

4.1  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 133]
- Access the My Inbox Application [page 134]

4.2  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.

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

**Note**
You can now view the entire information about the filters you are using on the task list in a tooltip.

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.

**Note**
When you release a task, you are no longer assigned as a processor of this task and it becomes visible in My Inbox for its other recipients. In this case, the status of the task changes from Reserved to Ready.

Execute and complete tasks.

### 4.2.1 Access 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 137]
- 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 Configuring SAP Fiori Launchpad Objects [page 16].
5 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

5.1 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 136].
• 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.

5.2 Identity Provider and Identity Management

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

All requests handled by the workflow service subscriptions are authenticated against the identity provider of the customer subaccount and authorized against the role assignments specified on the subscriptions in the customer subaccount. All users who need to interact with the various subservices of the workflow service must be available in the respective identity provider. You can replace the default 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 Authorization and Trust Management in the Neo Environment in the SAP Cloud Platform documentation.

Related Information

Groups in SAP Cloud Platform Workflow – Part 1

SAP Cloud Platform Workflow Security Guide
5.3 Authorization Configuration

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

Authorization for the workflow service is provided on two levels:

- Global authorization using the standard functionality of SAP Cloud Platform. Users with those platform roles assigned gain the associated permissions for all workflow definitions, instances, and tasks.
- Instance-specific authorizations using workflow service provided APIs. Users explicitly named by user ID, or as members of explicitly named groups, gain the associated permission for the respective workflow instance only. You can assign instance-specific permissions using the REST API. In addition, users need the platform WorkflowParticipant role.

Authorizations are cumulative: If any one authorization allows access, access is granted.

For more information about assigning global roles and permissions, see Configuring the Workflow Service [page 15].

Roles for Accessing Workflow Service Runtime

Global roles are marked and refer to the platform roles. Where available, instance-specific roles refer to the name used in the REST API.

Available Roles for Accessing the Workflow Service Runtime

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkflowDeveloper (global role)</td>
<td>- Permission to use the workflow editor&lt;br&gt;- Permission to query workflow definitions&lt;br&gt;- Permissions to retrieve the current error messages of a workflow instance</td>
</tr>
<tr>
<td>WorkflowContextAdmin (global role)</td>
<td>- Permissions to partially modify or completely override the workflow context&lt;br&gt;- Permission to retrieve the context of a task instance</td>
</tr>
<tr>
<td>contextAdminUsers</td>
<td></td>
</tr>
<tr>
<td>contextAdminGroups</td>
<td></td>
</tr>
<tr>
<td>WorkflowContextViewer (global role)</td>
<td>- Permission to retrieve the context of a workflow instance&lt;br&gt;- Permission to retrieve the context of a task instance</td>
</tr>
<tr>
<td>contextViewerUsers</td>
<td></td>
</tr>
<tr>
<td>contextViewerGroups</td>
<td></td>
</tr>
<tr>
<td>WorkflowInitiator (global role)</td>
<td>Permission to start workflow instances (using the API or the Monitor Workflows app)</td>
</tr>
<tr>
<td>Role</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WorkflowParticipant (global role)</td>
<td>• View tasks in My Inbox, where the user is a recipient</td>
</tr>
<tr>
<td></td>
<td>• Permissions to perform task operations:</td>
</tr>
<tr>
<td></td>
<td>○ claim</td>
</tr>
<tr>
<td></td>
<td>○ release</td>
</tr>
<tr>
<td></td>
<td>○ call the task completion API</td>
</tr>
<tr>
<td></td>
<td>○ eligible for having instance-specific permissions assigned</td>
</tr>
<tr>
<td>WorkflowAdmin (global role)</td>
<td>• Permission to use the Monitor Workflows app*</td>
</tr>
<tr>
<td>adminUsers</td>
<td>• Permissions to query workflow definitions as well as query and cancel workflow instances*</td>
</tr>
<tr>
<td>adminGroups</td>
<td>• Permissions to retrieve the tasks of a workflow instance</td>
</tr>
<tr>
<td></td>
<td>• Permissions to retrieve the current error messages of a workflow instance</td>
</tr>
<tr>
<td></td>
<td>• Permissions to retry the failed steps of an erroneous workflow instance</td>
</tr>
<tr>
<td></td>
<td>• Permissions to suspend and resume a workflow instance for temporary suspen­sion of processing</td>
</tr>
<tr>
<td></td>
<td>• Permissions to retrieve the workflow logs for a given workflow instance</td>
</tr>
<tr>
<td></td>
<td>• Permission to download the Workflow model in the Monitor Workflow app*</td>
</tr>
<tr>
<td>WorkflowMessageSender (global role)</td>
<td>Permission to send a message to a set of workflow instances for consumption in intermediate message events</td>
</tr>
<tr>
<td>WorkflowTenantOperator (global role)</td>
<td>• Permission to export data</td>
</tr>
<tr>
<td></td>
<td>• Permission to undeploy workflow definitions</td>
</tr>
<tr>
<td></td>
<td>• Permission to delete multiple workflow instances</td>
</tr>
<tr>
<td></td>
<td>• Permission to purge all workflow definitions, workflow instances, and form definitions.</td>
</tr>
<tr>
<td>WorkflowViewer (global role)</td>
<td>• Permissions to query workflow definitions* as well as query workflow instances</td>
</tr>
<tr>
<td>viewerUsers</td>
<td>• Permissions to retrieve the tasks of a workflow instance</td>
</tr>
<tr>
<td>viewerGroups</td>
<td>• Permissions to retrieve the workflow logs for a given workflow instance</td>
</tr>
<tr>
<td></td>
<td>• Permission to download the Workflow model</td>
</tr>
</tbody>
</table>

*) Only for global roles

**Related Information**

Access the My Inbox Application [page 134]
5.4 Destinations

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

The predefined destinations are generated and configured when enabling the workflow service in a customer subaccount. For more information, see Principal Propagation for User Interfaces [page 139] below.

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

- The workflow runtime uses destinations of type Mail to communicate with mail servers.
  For more information, see Configure the Workflow Service Mail Destination [page 19] and Configure Mail Tasks [page 65].
- To communicate with other services, the workflow runtime uses destinations of type HTTP.
  For more information, see Destination Configuration for Service Task [page 139] below and Configure Service Tasks [page 51]. For authentication and authorization purposes, also other, referenced destinations might be used, for example, OAuth2 authorization endpoints.

Principal Propagation for User Interfaces

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

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

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 Tasks

The workflow service supports outbound connectivity for orchestrating external services and systems. Destinations decouple modeling service tasks in your workflow model from the configuration of the physical back-end systems that are called in the service task at runtime.

→ Tip

Configure destinations to use secure communication protocols, such as HTTPS, wherever possible.

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

You can call services using the following authentication types:

- Basic Authentication: Select Basic Authentication as the authentication type in the destination.
- OAuth2 Client Credentials flow: For more information, see Neo Environment: Configure a Service Task Destination with OAuth2 Client Credentials Flow [page 140].
- OAuth2SAMLBearerAssertion: To propagate the user from certain actions on the workflow to other services, use this authentication type. For more information, see Configure a Service Task Destination with OAuth2SAMLBearerAssertion for Principal Propagation [page 141] and Configuring Principal Propagation for Service Tasks [page 21].

- No Authentication: If the service you want to call does not require any authentication, select No Authentication as the authentication type in the destination.

Besides the authentication type, the following destination features are supported in the workflow service:

- Server authentication (verification): JDK default and custom truststores
- Supported proxy type: Internet, OnPremise
- Destination type:
  - Supports HTTP and HTTPS connectivity based on HTTP destinations in SAP 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 about 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.

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

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

### 5.4.1 Neo Environment: Configure a Service Task Destination with OAuth2 Client Credentials Flow

**Prerequisites**

- Model a service task in a workflow model.
- Create a destination in your account where the name matches the destination property in the service task. In addition, make sure the URL points to the service you want to call.
- Verify that the service, which the service task should call, is protected with OAuth2 and supports the Client Credentials flow.
- Know the corresponding OAuth2 token endpoint from which a valid request token can be requested.
- Know the client ID and the secret for requesting access tokens.

**Procedure**

1. Configure the authentication type as *No Authentication*.
2. Create an additional property.
   
   Name the property `bpm.oauth.token.destination`. It must be able to use arbitrary values, for example, `ServiceTaskOAuthEndpoint`, and must point to a second destination.
3. Create a new destination with the name that you have specified for the property value in the previous step.
4. Configure the destination using the following data:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>An OAuth2 token endpoint from which a valid access token can be requested</td>
</tr>
<tr>
<td>Authentication type</td>
<td>Basic Authentication</td>
</tr>
<tr>
<td>User</td>
<td><code>&lt;client ID&gt;</code></td>
</tr>
<tr>
<td>Password</td>
<td>Set to client secret</td>
</tr>
</tbody>
</table>

5.4.2 Configure a Service Task Destination with OAuth2SAMLBearerAssertion for Principal Propagation

**Prerequisites**

- Model a service task in a workflow model.
- Create a destination in your account where the name matches the destination property in the service task. In addition, make sure the URL points to the service you want to call.
- Verify that the service, which the service task should call, is protected with OAuth2 and supports the OAuth2 SAML bearer assertion flow.
- Know the corresponding OAuth2 token endpoint URL, client ID, and client secret.
**Procedure**

1. Edit or create a destination as described in Configure Destinations from the Cockpit.
2. Select OAuth2SAMLBearerAssertion as the authentication type.
3. Configure the destination using the following data. Also maintain the additional properties necessary for your particular service provider. For more information, see SAML Bearer Assertion Authentication.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>audience</td>
<td>entityID property of the SAML 2.0 metadata.</td>
</tr>
<tr>
<td></td>
<td>For SAP Cloud Platform Neo, see Principal Propagation to OAuth-Protected Applications</td>
</tr>
<tr>
<td></td>
<td>For SAP Cloud Platform Cloud Foundry, see Principal Propagation from the Neo to the Cloud Foundry Environment.</td>
</tr>
<tr>
<td></td>
<td>To access a Neo service from the Cloud Foundry environment, you need to configure trust between the Cloud Foundry and Neo environments. See, for example, Principal Propagation from the Cloud Foundry to the Neo Environment.</td>
</tr>
<tr>
<td></td>
<td>For others, refer to the documentation of your service provider.</td>
</tr>
<tr>
<td>URL</td>
<td>Endpoint of the service you want to call</td>
</tr>
<tr>
<td>clientKey</td>
<td>Client ID of the OAuth client</td>
</tr>
<tr>
<td>tokenServiceURL</td>
<td>Token endpoint URL of the OAuth server</td>
</tr>
<tr>
<td>tokenServiceUser</td>
<td>Client ID of the OAuth client</td>
</tr>
<tr>
<td>tokenServicePassword</td>
<td>Client secret of the OAuth client</td>
</tr>
</tbody>
</table>

If you want to call a service in the Cloud Foundry environment, see the account ID in the cockpit in your space under | Instance | Service Key |. If you want to call a service in the Neo environment, see the account ID in the cockpit under | Security | OAuth | Clients |.

4. From the Additional Properties panel, choose New Property, and enter the following property:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>authnContextClassRef</td>
<td>urn:oasis:names:tc:SAML:2.0:ac:classes:X509</td>
</tr>
</tbody>
</table>

5. (Optional) If your destination points to a service in a different subaccount in the Neo or Cloud Foundry environment, you must configure trust between these accounts.
5.5 Data Protection and Data Privacy

Governments place legal requirements on industry to protect data and privacy. We provide features and functions to help you meet these requirements.

SAP does not provide legal advice in any form. SAP software supports data protection compliance by providing security features and data protection-relevant functions, such as blocking and deletion of personal data. In many cases, compliance with applicable data protection and privacy laws is not covered by a product feature. Furthermore, this information should not be taken as advice or a recommendation regarding additional features that would be required in specific IT environments. Decisions related to data protection must be made on a case-by-case basis, taking into consideration the given system landscape and the applicable legal requirements. Definitions and other terms used in this documentation are not taken from a specific legal source.

\[\text{Caution}\]

SAP Cloud Platform Workflow shall not be used for storing and processing sensitive personal data.

\[\text{Recommendation}\]

Working copies of data from systems of record that are stored in a workflow context should be limited to the very minimum required for the processing.

5.5.1 Information Report

An information report is a collection of data relating to a data subject. A data privacy specialist may be required to provide such a report or an application may offer a self-service.

REST API endpoints help the data privacy specialist when building a report. The data export endpoint, for example, enables the data privacy specialist to retrieve all relevant information for further processing.

For more information, see Using Workflow APIs [page 119].

5.5.2 Erasure

When handling personal data, consider the legislation in the different countries where your organization operates. After the data has passed the end of purpose, regulations may require you to delete the data.
However, additional regulations may require you to keep the data longer. During this period, you must block access to the data by unauthorized persons until the end of the retention period, when the data is finally deleted.

Personal data can also include referenced data. The challenge for deletion and blocking is first to handle referenced data and then other data, such as business partner data.

As part of the SAP Cloud Platform offboarding process, all data stored within the workflow service is deleted.

For audit needs, the workflow service offers an export feature. For more information, see Export Workflow Service Data [page 30].

To delete workflow definitions, workflow context data, or form definitions, the workflow service provides REST APIs. For more information, see the SAP Cloud Platform Workflow API

⚠️ Caution

Workflow definitions and form definitions are persisted separately. Deleting a workflow definition does not delete dependent form definitions and the other way round.

Deleting dependent artifacts of a workflow, such as form definitions, may break existing workflow definitions and running workflow instances.

5.5.3 Change Log

For auditing purposes or for legal requirements, changes made to personal data should be logged, enabling the monitoring of who made changes and when.

Therefore, SAP Cloud Platform Workflow may write logs into the audit log handled by the platform itself.

ℹ️ Note

SAP Cloud Platform Workflow does not provide inherent support for logging changes in the workflow context.

The workflow developer must take care of logging changes to attributes in the workflow context that hold personal data. Such changes may occur, for example, when calling external services, through intermediate message events, or when updating context data through the REST API.

Workflow definitions may include personal data, for example, the user IDs of task recipients. For this kind of data, the API provides versioning access at /v1/workflow-definitions/{definitionId}/versions.

The workflow service contains information about which users completed which tasks. You can retrieve this information using the REST API endpoint /v1/workflow-instances/{workflowInstanceId}/execution-logs.

Furthermore, it contains information about which user has deployed a form definition. You can retrieve this information by using the data export endpoint, see Information Report [page 143].
## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocking</td>
<td>A method of restricting access to data for which the primary business purpose has ended.</td>
</tr>
<tr>
<td>Business purpose</td>
<td>A legal, contractual, or in other form justified reason for the processing of personal data. The assumption is that any purpose has an end that is usually already defined when the purpose starts.</td>
</tr>
<tr>
<td>Consent</td>
<td>The action of the data subject confirming that the usage of his or her personal data shall be allowed for a given purpose. A consent functionality allows the storage of a consent record in relation to a specific purpose and shows if a data subject has granted, withdrawn, or denied consent.</td>
</tr>
<tr>
<td>Deletion</td>
<td>Deletion of personal data so that the data is no longer available.</td>
</tr>
<tr>
<td>End of business</td>
<td>Date where the business with a data subject ends, for example the order is completed, the subscription is canceled, or the last bill is settled.</td>
</tr>
<tr>
<td>End of purpose (EoP)</td>
<td>End of purpose and start of blocking period. The point in time, when the primary processing purpose ends (for example contract is fulfilled).</td>
</tr>
<tr>
<td>End of purpose (EoP)</td>
<td>End of purpose and start of blocking period. The point in time, when the primary processing purpose ends (for example contract is fulfilled).</td>
</tr>
<tr>
<td>End of purpose (EoP) check</td>
<td>A method of identifying the point in time for a data set when the processing of personal data is no longer required for the primary business purpose. After the EoP has been reached, the data is blocked and can only be accessed by users with special authorization (for example, tax auditors).</td>
</tr>
<tr>
<td>Personal data</td>
<td>Any information relating to an identified or identifiable natural person (&quot;data subject&quot;). An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural, or social identity of that natural person</td>
</tr>
<tr>
<td>Purpose</td>
<td>The information that specifies the reason and the goal for the processing of a specific set of personal data. As a rule, the purpose references the relevant legal basis for the processing of personal data.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Residence period</td>
<td>The period of time between the end of business and the end of purpose (EoP) for a data set during which the data remains in the database and can be used in case of subsequent processes related to the original purpose. At the end of the longest configured residence period, the data is blocked or deleted. The residence period is part of the overall retention period.</td>
</tr>
<tr>
<td>Retention period</td>
<td>The period of time between the end of the last business activity involving a specific object (for example, a business partner) and the deletion of the corresponding data, subject to applicable laws. The retention period is a combination of the residence period and the blocking period.</td>
</tr>
<tr>
<td>Sensitive personal data</td>
<td>A category of personal data that usually includes the following type of information:</td>
</tr>
<tr>
<td></td>
<td>- Special categories of personal data, such as data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, trade union membership, genetic data, biometric data, data concerning health or sex life or sexual orientation.</td>
</tr>
<tr>
<td></td>
<td>- Personal data subject to professional secrecy</td>
</tr>
<tr>
<td></td>
<td>- Personal data relating to criminal or administrative offenses</td>
</tr>
<tr>
<td></td>
<td>- Personal data concerning insurances and bank or credit card accounts</td>
</tr>
<tr>
<td>Where-used check (WUC)</td>
<td>A process designed to ensure data integrity in the case of potential blocking of business partner data. An application's where-used check (WUC) determines if there is any dependent data for a certain business partner in the database. If dependent data exists, this means the data is still required for business activities. Therefore, the blocking of business partners referenced in the data is prevented.</td>
</tr>
</tbody>
</table>
6 Troubleshooting

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

**Note**

We recommend that you also check the Questions & Answers for SAP Cloud Platform Workflow.

If none of these resources helps solve your problem, please open a ticket. See SAP Note 1888290.

**Related Information**

- End Users Can't Open SAP Fiori Launchpad Tiles [page 147]
- Error When Clicking "Go to Service" on Portal Tile [page 152]
- HTTP Status 403: User Doesn't Have Sufficient Privileges [page 153]
- Tasks Not Appearing in My Inbox [page 153]
- Error During Workflow Deployment in SAP Web IDE [page 154]
- No Permissions Granted [page 154]

### 6.1 End Users Can't Open SAP Fiori Launchpad Tiles

**Symptom**

**Note**

This feature is not available with the Beta of workflow service in the Cloud Foundry environment.

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.*
See the list of possible causes for the failure and the respective solutions:

- 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. To clear the cache, see Clear the SAP Fiori Launchpad Cache [page 148].
- The SAPUI5 version on SAP Fiori launchpad is not the latest version. To update the version, see Change the SAPUI5 Version [page 149].

Result

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

6.1.1 Clear the SAP Fiori Launchpad Cache

You use this procedure to resolve the cache issues in SAP Fiori launchpad.

Prerequisite

You are unable to open the SAP Fiori launchpad tiles. For more information, see End Users Can’t Open SAP Fiori Launchpad Tiles [page 147].

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.”

### 6.1.2 Change the SAPUI5 Version

You use this procedure to change the SAP Fiori launchpad to the latest version.

**Prerequisite**

You are unable to open the SAP Fiori launchpad tiles. For more information, see [End Users Can't Open SAP Fiori Launchpad Tiles](#page_147).

**Solution**

You have the 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**.

6. In the **System Settings** section, ensure that the **SAPUI5 Version** is set to **Latest supported version**.

7. If the version is not set to the latest, then choose **Edit**.
8. From the SAPUI5 Version list, choose Latest Supported Version.

9. Choose Save.

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

6.2 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.
6.3  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 137].
- Assign the Workflow Participant role to your user and all other users that are supposed to access My Inbox.

Check whether the workflow roles are assigned to you:
1. Log on to the SAP Cloud Platform cockpit.
2. Access your subaccount and choose Services from the navigation area.
3. Search for the workflow service.
5. In the navigation area, choose Roles, and assign the necessary roles to your user.
   For more information on roles and permissions, see Authorization Configuration [page 137] and HTML: Workflow Service API Reference.
6. Log off and log on again for the changes to take effect.

Related Information

Tutorial Step 5: Assign roles

6.4  Tasks Not Appearing in My Inbox

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.

Related Information

Groups in SAP Cloud Platform Workflow – Part 1
6.5 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:

Solution

See the list of possible causes for the failure and the respective solutions:

- SAP Cloud Platform Workflow isn’t enabled. To enable the workflow service, see Configuring the Workflow Service [page 15].
- The destination isn’t configured correctly. To configure the destination, perform the following steps:
  1. Log in to your subaccount in the SAP Cloud Platform cockpit.
  2. In the navigation area, choose Connectivity ➤ Destinations ➤
  3. Choose the bpmworkflowruntime destination.
  4. In the Destination Configuration section, choose Edit.
  5. In Additional Properties, select New Property, then choose WebIDEEnabled as the new property.
  6. Set this property to true.
  7. Choose Save.

Result

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

6.6 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.

6.7   Workflow Service Cannot be Enabled in the Account

Symptom

In the cockpit, enabling the workflow service fails.

Solution

1. Check whether the Portal service is enabled.
2. Retry enabling the workflow service.
3. Check the Questions & Answers.
4. Open a ticket as described in https://launchpad.support.sap.com/#/notes/1888290.

6.8   Category for Web IDE Template Not Available

Symptom

You want to create a workflow project from a template in SAP Web IDE Full-Stack but the Category field does not offer the entry Business Process Management.

Solution

1. Leave the template wizard.
2. Choose Tools ▶ Preferences ▶ Features.
3. Search for "workflow".
4. Enable the feature.
5. Choose Save.

6.9   Cannot Deploy Workflow

Symptom

i Note

This feature is not available with the Beta of workflow service in the Cloud Foundry environment.
You created a workflow in the workflow editor in SAP Web IDE Full-Stack. When you try to deploy the workflow, you are not allowed to do so.

**Solution**

Check whether the workflow roles are assigned to you:

1. Log on to the SAP Cloud Platform cockpit.
2. Access your subaccount and choose Services from the navigation area.
3. Search for the Workflow service.
5. In the navigation area, choose Roles, and assign the WorkflowDeveloper role to your user.
6. Log off and log on again for the change to take effect.

### 6.10 Service Calls Fail

**Solution**

1. In the Monitor Workflow - Workflow Instances app, look for errors under ERROR MESSAGES.
2. Check whether the destination exists and the path is valid.
3. If basic authentication is used, check whether the user and password are valid.
4. On-premise destination and cloud connector: Make sure the domain mapping is correct. For more information, see this answer.
5. If principal propagation is configured for the service task, verify the following:
   - Verify that you have executed all steps for enabling principal propagation as described in Configuring Principal Propagation for Service Tasks [page 21].
   - Check whether you have configured the service task destination as described in Configure a Service Task Destination with OAuth2SAMLBearerAssertion for Principal Propagation [page 141].
   - If you have modified the role or group assignments for the user who is propagated to the service task, but if those changes are not reflected in the service task call:
     - Make sure that the affected user has logged out and logged in again.
     - Wait until the token for the OAuth client, which is configured in the service task destination, has expired. The expiration depends on the token lifetime, which is configured in the OAuth client.
     - Make sure that the affected user has either completed a user task or started a workflow, where the user task or start event is configured as principal propagation source.

**Related Information**

Managing Task Instances [page 25]
6.11  My Inbox Features Not Active

Symptom

You are used to work with My Inbox and are missing features usually available there.

Solution

Some features are generally available for My Inbox, but not all of them are already available for My Inbox used by the workflow service.

Related Information

Working with Tasks in My Inbox [page 133]

6.12  Authentication Issues when Using a Custom IDP and Basic Authentication

Symptom

You use a custom IDP and experience authentication issues when accessing the workflow service using basic authentication (HTML response instead of a successful authentication).

Solution

Your custom IDP is not enabled for basic authentication by default. To change this configuration, create a support ticket on component BC-NEO-SEC-IAM.
7 2017 - What's New for Workflow Service

19 December 2017 (CW51) - Workflow Service

- Enhanced
  - Minor updates and bugfixes.

12 December 2017 (CW50) - Workflow Service

- Enhanced
  - Using the REST API, when searching for workflow instances, the count of all found instances can be returned.
  - For more information, see SAP Cloud Platform Workflow API.

- Enhanced
  - Every entry of the Workflow Instances in the Manage Workflows app has a Refresh button.
  - For more information, see Managing Workflows Using the Monitor Workflows App [page 23].

5 December 2017 (CW49) - Workflow Service

- Enhanced
  - Using the REST API, the following task properties can be updated:
    - Processor
    - Recipient
  - For more information, see SAP Cloud Platform Workflow API and Updating Task Properties [page 127].

- New
  - You can format your workflow model using the Arrange Horizontally and Arrange Vertically options.
  - For more information, see
    - Editor Layout [page 36]
    - Define Workflows [page 37]
Enhanced

Using the REST API, the following task properties can be updated:

- Due date
- Priority
- Description
- Subject

For more information, see SAP Cloud Platform Workflow API and Updating Task Properties [page 127].

Enhanced

In My Inbox, you can sort by the user who created a task. This information is derived from the initiator of the workflow instance by which the task was created. It is also available in the workflow service API.

For more information, see SAP Cloud Platform Workflow API.

21 November 2017 (CW47) - Workflow Service

Enhanced

Minor user interface enhancements in the editor layout.

For more information, see Editor Layout [page 36]

13 November 2017 (CW46) - Workflow Service

New

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.

For more information, see:

- Configure User Tasks [page 39]
- Working with Tasks in My Inbox [page 133]
**Enhanced**

Minor user interface changes while configuring the timer events.

For more information, see:

- Configure Intermediate Timer Events [page 71]
- Configure Boundary Timer Events [page 42]

---

**7 November 2017 (CW45) - Workflow Service**

**Enhanced**

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**.

For more information, see Managing Workflows Using the Monitor Workflows App [page 23].

---

**Enhanced**

In the My Inbox, end users can see and filter using the task names that are defined in the editor.

For more information, see Working with Tasks in My Inbox [page 133].

---

**Enhanced**

In the My Inbox, users can group tasks by task types.

For more information, see Working with Tasks in My Inbox [page 133].

---

**17 October 2017 (CW42) - Workflow Service**

**New**

Clients have the option to also authenticate using OAuth 2.0 when using workflow service APIs.

For more information, see Access Workflow APIs Using OAuth 2.0 Authentication (Client Credentials) [page 120] and Access Workflow APIs Using OAuth 2.0 Authentication (Authorization Grant) [page 122].

---

**Change**

You can create JavaScript files directly from **Script Task Properties**.

For more information, see Configure Script Tasks [page 56].
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.

For more information, see Configure User Tasks [page 39].

You can configure the user interface properties in a user task using a lookup provided for the SAPUI5 projects in your workspace.

For more information, see Configure User Tasks [page 39].

You can export runtime data related to workflow definitions and workflow instances. You can use this data to address, for example, audit needs.

For more information, see Export Workflow Service Data [page 30].

In the REST API, task-related execution log entries contain the task instance ID of the respective task.

You can model an intermediate timer event that allows you to pause and resume after a specified interval of time.

For more information, see Configure Intermediate Timer Events [page 71].

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.

For more information, see Configure Boundary Timer Events [page 42].
19 September 2017 (CW38) - Workflow Service

New
The workflow deployment errors are displayed in the Problems view of SAP Web IDE Full-Stack.
For more information, see Deploy Workflows [page 75].

New
You can configure the My Inbox application to consume tasks from another TCM-compliant OData Provider.
For more information, see Configure My Inbox to Consume Tasks from Another TCM-Compliant OData Provider [page 18].

Enhanced
The Properties pane in the Workflow editor is vertically aligned.

12 September 2017 (CW37) - Workflow Service

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 Managing Workflows Using the Monitor Workflows App [page 23].

5 September 2017 (CW36) - Workflow Service

New
Workflow errors are displayed in the Problems view of SAP Web IDE Full-Stack.
For more information, see Define Workflows [page 37].

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 56].
29 August 2017 (CW35) - Workflow Service

Enhanced

Minor updates and bugfixes.

22 August 2017 (CW34) - Workflow Service

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 124]
- Using Workflow APIs [page 119]

15 August 2017 (CW33) - Workflow Service

New

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.

For more information, see:

- Configure Intermediate Message Events [page 69]
- Define Workflows [page 37]
- Accelerated Modeling with Speed Buttons [page 76]

New

With the REST API, you can send message events to specific workflow instances.

For more information, see

- Authorization Configuration [page 137]
- Workflow Execution Log [page 130]

Enhanced

You can use not only concrete user assignments for user tasks, but also group assignments, containing multiple users.

For more information, see Configure User Tasks [page 39].
8 August 2017 (CW32) - Workflow Service

**New**
You can claim and release tasks in My Inbox.

**Enhanced**
With the REST API, you can suspend and resume a workflow instance.
For more information, see Using Workflow APIs [page 119].

**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

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

18 July 2017 (CW29) - Workflow Service

**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 37] (SAP Web IDE workflow editor) and the standalone workflow editor.

**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

New
You can model workflows using the workflow editor feature in SAP Web IDE.
For more information, see Modeling a Workflow [page 33].

11 July 2017 (CW28) - Workflow Service

Enhanced
Minor updates and bugfixes.
For more information, see Workflow Service API documentation.

4 July 2017 (CW27) - Workflow Service

Enhanced
With the REST API or the Monitor Workflows app, workflow administrators have the following options:

- Page workflow instances
- Use free-text backend search
For more information, see Using Workflow APIs [page 119].

Enhanced
With the REST API workflow administrators can filter workflow instances.
For more information, see Using Workflow APIs [page 119].

Enhanced
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.
For more information, see Using Workflow APIs [page 119].

Enhanced
For each user task you can decide whether to display the respective execution log information in the inbox.
27 June 2017 (CW26) - Workflow Service

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

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.

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

13 June 2017 (CW24) - Workflow Service

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 Managing Workflows Using the Monitor Workflows App [page 23].

7 June 2017 (CW23) - Workflow Service

Enhanced
Minor runtime bugfixes
30 May 2017 (CW22) - Workflow Service

New
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.

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 Workflow Started entry. An incomplete history has no impact on the execution of the workflow.

For more information, see Using Workflow APIs [page 119] and Workflow Execution Log [page 130].

23 May 2017 (CW21) - Workflow Service

Enhanced
Minor documentation corrections

16 May 2017 (CW20) - Workflow Service

Enhanced
Minor runtime bugfixes

9 May 2017 (CW19) - Workflow Service

New
You can navigate to a section of the workflow using the diagram overview in the workflow service editor.

Enhanced
The speed buttons available while modeling a workflow are grouped together under tasks and gateways.
For more information, see Accelerated Modeling with Speed Buttons [page 76].

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

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 139].

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 119].

4 April 2017 (CW14) - Workflow Service

Enhanced

You can use context variables using JUEL expressions in a unified way in the workflow service editor.

For more information, see Expressions [page 77].
Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.

About the icons:

- Links with the icon 🌐: You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP’s gross negligence or willful misconduct.
- Links with the icon 🛡️: You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.

The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP’s gross negligence or willful misconduct.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.