Users Guide

Generated on: 2023-04-20 11:01:40 GMT+0000

SAP Fiori tools | Current

PUBLIC

Original content: https://help.sap.com/docs/SAP_FIORI_tools/17d50220bcd848aa854c9c182d65b699?locale=en-US&state=PRODUCTION&version=Latest

Warning

This document has been generated from the SAP Help Portal and is an incomplete version of the official SAP product documentation. The information included in custom documentation may not reflect the arrangement of topics in the SAP Help Portal, and may be missing important aspects and/or correlations to other topics. For this reason, it is not for productive use.

For more information, please visit the https://help.sap.com/docs/disclaimer.

Develop SAP Fiori Applications with SAP Fiori Tools

To develop an application that connects to SAP data sources, you can use one of the following options:

- SAP Fiori elements creates standard applications based on several basic floorplanss for common business scenarios. These floorplans provide UI code, including some logic, so that the data from the backend goes exactly where it's supposed to, and the UI behavior is based on the metadata you provide. You can customize the floorplans using XML Annotations.
- SAPUI5 freestyle gives you complete flexibility about your application's look and performance. You can select floorplans and specify the layout, flow, menu structure, colors, fonts, interactions, patterns, and more. For each screen, you have to write the JavaScript UI code, which may be time-consuming.

What Is SAP Fiori Elements?

SAP Fiori elements is one of the options available to develop SAP Fiori apps based on SAPUI5. SAP Fiori elements provide designs for UI patterns and predefined templates for common application use cases. Using standard templates as the starting point for creating applications increases developer efficiency, enforces UX consistency across apps, and ensure that applications created using SAP Fiori elements include features expected in a typical enterprise.

Application developers can use SAP Fiori elements to create SAP Fiori apps based on an OData service and annotations that do not need JavaScript UI coding to create a front end of your app. This means that no application-specific view instances are required. The SAP Fiori elements runtime interprets the metadata and annotations of the underlying OData service and uses the corresponding views for the SAP Fiori apps at startup. SAP Fiori apps are created with SAPUI5 and follow the SAP Fiori design guidelines to ensure consistency and high quality level. See SAP Fiori Design Guidelines.

For more information about SAP Fiori apps, see http://www.sap.com/fiori.

For more information about SAP Fiori, see SAP Fiori Overview.

For more information about SAP Fiori elements, see Developing Apps with SAP Fiori elements.

Project Structure and Artifacts of SAP Fiori elements

The SAP Fiori elements framework interprets OData metadata and annotations to render the application UI. While metadata describes your data model or what is on your screen, the annotations describe your data semantics or the way it's visualized.

For example, any entity type property that has the `Communication.IsEmailAddress` annotation term set to true is interpreted by the SAP Fiori elements framework as an e-mail address and displayed in an application UI as a clickable link that leads to the e-mail client.

Annotations can be associated to **entities**, **relationships**, and **properties** of a service.

You can also use annotations to perform the following actions:

- Describe the relationships between **properties**, such as between an amount and the related currency. SAP Fiori elements display these **properties** side by side in an application UI.
- Group **properties** together so that they are displayed next to each other in a form.
- Describe the actions available on a given object, whether you can edit it, delete it, apply some filters, and more.

All of this is interpreted by the SAP Fiori elements framework and shown in the application accordingly.

Annotations can be defined together with the metadata or in dedicated annotation files, both in the backend along with the service and in the frontend, in the local annotation files.

For an annotation to be considered by the SAP Fiori elements framework during runtime, it must reside in an annotation source registered in an application descriptor file `manifest.json` of the application.

Additionally, in the web app manifest, the description of your application basic metadata can be found, such as its name and version, as well as its content, such as application pages, navigation, service details used, and more.

The same descriptor file `manifest.json` that is a runtime source for your annotations can be used for the following operations:

- Adding navigation from pages, such as a drill down from a **List Report** to an **Object Page**.
- Define the settings of tables, columns, or other UI elements. In addition, registering custom extensions like custom columns, pages or sections.

For flex changes in the individual pages, such as enabling export, you can use the **Page Editor**.

What Is SAPUI5 Freestyle?

SAPUI5 freestyle is another option available to develop SAP Fiori apps. The user can choose the type of template required, along with the relevant OData service to start your application development process. By using the SAPUI5 freestyle templates, you can be flexible in creating a custom application based on an OData service and JavaScript UI coding. SAPUI5 freestyle incorporates our latest recommendations and can be used as a starting point for developing custom apps according to the SAP Fiori design. See SAP Fiori Design Guidelines.

For more information on SAP Fiori apps, see http://www.sap.com/fiori.

For more information on SAPUI5, see Get Started with UI5.

Project Structure and Artifacts of SAPUI5 Freestyle

With the SAPUI5 freestyle framework, you can optimize your SAP Fiori-based development by using OData and SAPUI5 tooling to render the application UI. The SAPUI5 controls are convenient and intuitive to use for your app development. Additionally, they ensure adherence to the design guidelines.

The SAPUI5 freestyle templates create at least three basic files for your application:

- `manifest.json`
- `app.view.xml`
- `component.js`

With each of these files, you can control certain aspects of your application. For more information, see Basic Files for your App.

ℹ Note

The SAPUI5 freestyle templates are deprecated, and it's recommended to use the custom page SAP Fiori template based on the flexible programming model as an alternative. For more information, see Flexible Programming Model.

What Is the Difference between SAP Fiori Elements and SAPUI5 Freestyle?

The following table lists the differences between SAPUI5 freestyle and SAP Fiori elements.

	SAPUI5 freestyle	SAP Fiori elements
OVERALL APPROACH	Flexibility	**Efficiency**
DESIGN REQUIREMENTS	Freestyle designs	SAP templates
DEVELOPMENT KNOWLEDGE	Web development	Annotations
OWNERSHIP & MAINTENANCE	Own coding, own UI logic	Own annotations, SAP's UI logic
TOTAL COST	Higher	**Lower**

Create SAP Fiori Application with SAP Fiori Tools

SAP Fiori tools provides many capabilities to increase the efficiency of developing SAP Fiori applications using either SAP Fiori elements or SAPUI5 freestyle approach. SAP Fiori tools, together with SAP Fiori elements, reduce development time, maintenance cost, and leverage the advantages of a metadata driven UI. SAP Fiori tools includes the following:

- Wizard for the initial creation of an application.
- Service modeler for viewing the data model.
- XML and form-based editor for maintaining of annotations - SAP Fiori elements only.
- Application page structure and ability to configure the SAPUI5 flexibility settings - SAP Fiori elements only.
- Guided Development for implementing features - SAP Fiori elements only.

See Getting Started with SAP Fiori Tools on how to use SAP Fiori tools.

Getting Started with SAP Fiori Tools

SAP Fiori tools simplifies the development of SAP Fiori elements applications by providing extensions for your SAP Business Application Studio and VS Code development environment.

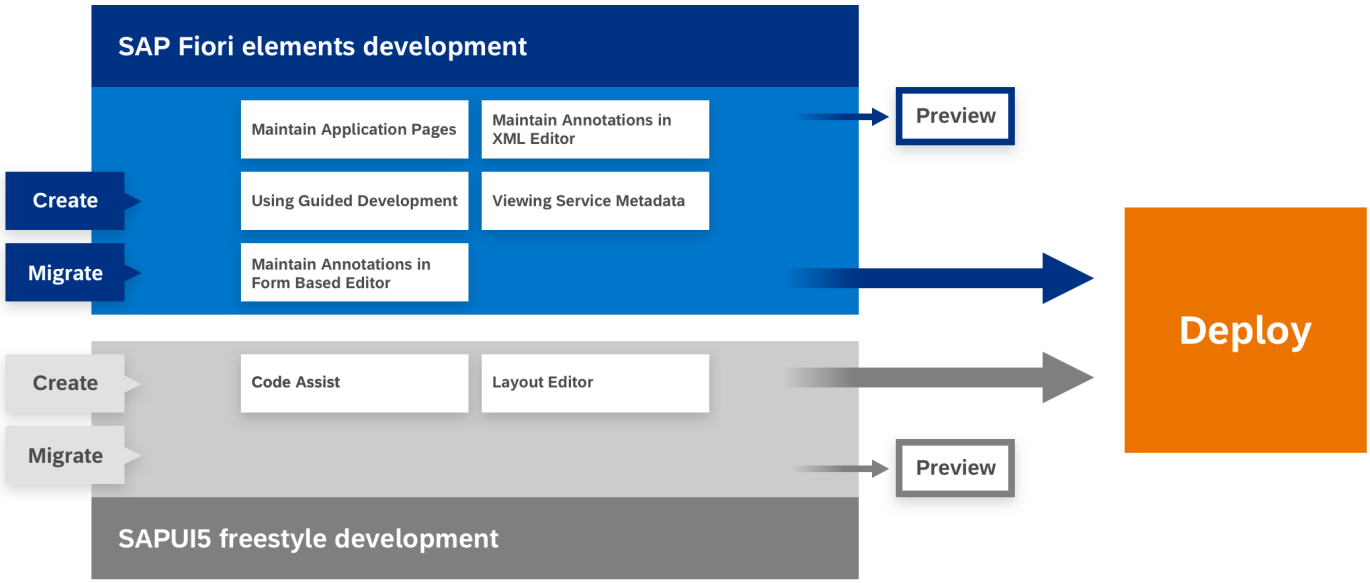
The SAP Fiori tools extensions help you create applications, visualize navigation, automatically generate code, and more. Used in combination with SAP Fiori elements, these extensions can increase your development productivity, ensure the consistency of experience between applications, and help you build a scalable experience. SAP Fiori tools include the following extensions:

- Application Wizard. A wizard-style approach to generate the provided SAP Fiori elements and SAPUI5 freestyle templates.
- SAP Fiori Tools – Application Modeler. Access to a visualization of the application pages, navigation, and service entities. You can add new navigation and pages, delete pages, and navigate to corresponding editing tools. The following features are part of this extension:
  - Page Editor

    - Page Map

- Guided Development. Access to **How-To** guides and tutorials that explain how to implement certain functionality in an SAP Fiori elements application. You can follow the steps required to implement a feature and then use the guided development approach to make the required changes in your project.
- Service Modeler. Visualization of the OData service metadata files. You can use it to browse complex services easily, including entities, properties, and associations.
- Maintaining Annotations with Language Server. Access to resources that help to define annotations in the code editor, thus improving application development by reducing effort and maintaining code consistency. The following subset of features is part of this extension:
    - Code completion
    - Micro-snippets
    - Diagnostics
    - Internationalization support
- SAP Fiori tools Environment Check. A tool, which is designed to run checks against your environment and/or destinations configured in SAP BTP.

This image is interactive. Hover over each area for a description. Click highlighted areas for more information.



Please note that image maps are not interactive in PDF output.

See Report Issues and Security for more information.

Supported Deployment Landscapes:

- SAP S/4HANA
- SAP S/4HANA Cloud
- SAP BTP (ABAP and Cloud Foundry Environments)

For supported data sources see: Data Source

ℹ Note

In case of any issues, create an incident in the **SAP Support Portal** for component **CA-UX-IDE**.

SAP Fiori tools is available in two IDEs

- SAP Business Application Studio
- Visual Studio Code

⚠ Caution

Currently, SAP Fiori tools supports development of SAP Fiori elements and SAPUI5 freestyle applications with minimum SAPUI5 versions 1.65 or higher.

Installation

This section contains installation instructions on the following environments:

- SAP Business Application Studio
- Visual Studio Code

SAP Business Application Studio

SAP Business Application Studio, also known as SAP BAS, is a hosted environment, which does not require installation. To create your workspace with BAS, perform the steps defined in the Set Up SAP Business Application Studio for Development tutorial. We recommend that after this tutorial you proceed with the next one in this group, which is Set Up and Generate a New SAP Fiori App Project.

Installing Extensions and Dev Space in SAP Business Application Studio

SAP Fiori tools are preinstalled and loaded when the dev space is created in SAP Business Application Studio for the following dev space types:

- **SAP Fiori Dev Space**
- **Full Stack Cloud Application Dev Space**. This dev space is intended for services and applications that using SAP Cloud Application Programming Model (CAP).
- In SAP Business Application Studio, an **SAP Fiori tools Dev Space** does not include the CDS OData Language Server extension. For more information, see **SAP CDS Language Support** in the Visual Studio Code section.
- In SAP Business Application Studio, an **SAP Fiori tools Dev Space** already includes UI5 Language Assistant as it is a mandatory extension installed automatically.

  For more information, see UI5 Language Assistant Support .

Related Information

Set Up SAP Business Application Studio for Development
SAP Business Application Studio - Getting Started

Visual Studio Code

This chapter describes how you can install and start working with Microsoft Visual Studio Code (also known as VS Code) and provides links to important step-by-step procedures.

System Requirements

Ensure that the minimum system requirements for installing VS Code are met. For more information, see Requirements for Visual Studio Code .

- Before VS Code installation, you must have Node.js installed. Ensure you're using version 16.18 or higher of Node.js. Furthermore, the version of Node.js you install must also have the corresponding version of npm installed. See Node.js releases for details on the versions of Node.js marked as LTS, and their associated npm versions.

  → Tip

  To check the version of Node.js installed, type `node -v` in the terminal. Similarly, to check the version of npm installed, type `npm -v` in the terminal. Ensure that the npm version and Node.js version are compatible.

  You can download Node.JS here: https://nodejs.org/en/download/ .

  ⓘ Note

  For Mac OS, we don't recommend installing Node.js from the executable download. It's recommended to install Node.js with **Home Brew** .

  Installing Node.js with Home Brew (Mac OS only)

  To install Node.js with Home Brew , perform the following steps:

  1. In the terminal, at the command line, enter the following command to install **Home Brew** :

     ```
     /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/insta
     ```

  2. Install the required version of Node.js by entering the following command at the command line:

     ```
     brew install node@16
     ```

     Where 16 refers to the version of Node.js you would like to install.

  3. Before continuing with other installation steps using terminal, close and start a new terminal.

- The SAP Fiori application generator requires the MTA tool Node.js package (version 1.0 or higher) to be installed globally.

  At the command line, enter the following command to install MTA:

  ```
  npm install -g mta
  ```

Cloud Foundry CLI tools

To access Cloud Foundry services from SAP Business Technology Platform, download and install the latest version of Cloud Foundry Command Line Interface (CF CLI) by following the installation instructions .

Prerequisites

Ensure that you have the following scope set in your `npm` configuration. Execute the following command:

```
npm config get @sap:registry
```

One of the following values should be returned:

- https://registry.npmjs.org
- undefined

If it is set incorrectly to the `@sap`, open your **`.npmrc`** file in your home directory and remove this entry.

Set up Visual Studio Code

To set up VS Code, you need to perform the following steps:

1. Download VS Code from the Visual Studio Code website .

   ℹ Note

   To learn how to get started with VS Code, see https://code.visualstudio.com/docs/setup/setup-overview .

2. You must have a working knowledge of VS Code.

   Using VS Code requires a working knowledge of this environment. We encourage you to use the following resources to obtain answers to your VS Code questions and keep yourself informed:

   - Visual Studio Code Basic Layout .
   - Visual Studio Code Introductory Videos .

3. Check if `Node JS` is already installed. You can check this by executing the following in the VS Code terminal. If you don't get a version number, see above on how to install `Node JS`.

   ```
   node -v
   ```

Install Extensions

1. Navigate to **VS Code** > **Extensions**.

   For more information, see Install Visual Studio Code extensions .

2. Search for SAP Fiori tools and select **SAP Fiori tools - Extension Pack**.

   Alternatively, you can navigate to SAP Fiori tools - Extension Pack and click Install.

3. Click Install.

4. Then, the SAP Fiori tools install the latest release of the following extensions:

   - Application Wizard
   - SAP Fiori Tools – Application Modeler
   - SAP Fiori tools - Guided Development
   - SAP Fiori tools - Service Modeler

- SAP Fiori tools - XML Annotation Language Server
- XML Toolkit

You can also install each extension separately.

> **Note**
>
> SAP CDS Language Support is an optional extension that can be installed for annotation LSP in CAP CDS. For more information, see CDS Editor.

SAP CDS Language Support

For applications based on CAP, you can install **SAP CDS Language Support** extension. To do so, perform the following steps:

1. Open SAP CDS Language Support extension page in Visual Studio Marketplace.
2. Click Install to open a new page with the **SAP CDS Language Support** extension in VS Code.
3. On the VS Code page, click **Install** to enable the extension.

For more information, see Add CDS Editor and CDS Editor.

UI5 Language Assistant Support

UI5 Language Assistant Support is an openSource extension that can be optionally installed to perform control ID checks when *flexEnabled* property is set to true in the `manifest.json` file for either SAP Fiori elements or SAPUI5 freestyle projects. It also provides additional support for relevant filters to suggestions and text for ease of use.

To install the UI5 Language Assistant Support extension or read about what features it supports, click **Install** in the UI5 Language Assistant VS Code marketplace page.

Authentication Type Support

The following are supported authentication types with SAP Fiori tools running in VS Code.

Authenticatio Type	SAP on-Premise	SAP BTP, ABAP Environment	SAP BTP, Cloud Foundry	SAP S/4 HANA Cloud
OAuth 2.0 (Client Credentials)	No	Yes	No	No
Basic Authentication	Yes	No	Yes	No

We recommend using SAP Business Application Studio for the extensive support of the authentication types especially when your required authentication type that is not supported in VS Code. Please refer to this Note:0002577263 for more information.

If applicable, disable `SAML` for selected `OData` services. Below are the list of `OData` services:

OData Services	Path
OData V2 catalog	/sap/opu/odata/IWFND/CATALOGSERVICE;v=2
OData V4 catalog (dev)	/sap/opu/odata4/iwfnd/config/default/iwfnd/catalog/0001
OData V4 catalog (prod)	/sap/opu/odata4/iwfnd/config/default/iwfnd/catalog/0002

OData Services	Path
ATO Catalog	/sap/bc/adt/ato/settings
SAPUI5 repository service (for deploy & undeploy)	/sap/opu/odata/UI5/ABAP_REPOSITORY_SRV

Feature Matrix

ℹ Note

To view information on features available in your application, navigate to **Command Palette** > **Fiori: Open Application Info** in VS Code or SAP Business Application Studio. For more details, see Application Information.

The following table lists the SAP Fiori tools features provided in each template or data source.

Feature	SAP Fiori elements	SAPUI5 freestyle	ODataV2	OData V4	EDMX - ABAP-based service	CAP - Node.js	CAP - Java
Preview Application	✓	✓	✓	✓	✓	✓	✓
Validate Application	✓	✓	✓	✓	✓		
Check Service	✓		✓	✓	✓		
Manage Service Models	✓	✓	✓	✓	✓		
Manage XML Annotations	✓	✓	✓	✓	✓	✓	✓
Open Guided Development	✓		✓	✓	✓	✓	✓
Open Page Map	✓		✓	✓	✓	✓	✓
Build Application	✓	✓	✓	✓			
Add Deploy Config	✓	✓	✓	✓	✓	✓	
Deploy Application	✓	✓	✓	✓	✓	✓	
Undeploy Application	✓	✓	✓	✓	✓	✓	
Check Node Modules	✓	✓	✓	✓	✓	✓	✓
Maintain Mockdata	✓	✓	✓	✓	✓		
Create Archive	✓	✓	✓	✓			

Feature	SAP Fiori elements	SAPUI5 freestyle	ODataV2	OData V4	EDMX - ABAP-based service	CAP - Node.js	CAP - Java
Show Documentation	✓		✓	✓	✓	✓	✓
Change minUI5 Version	✓	✓	✓	✓	✓	✓	✓
Add Mockserver Config	✓	✓	✓	✓	✓		

Usability

The following support keyboard navigation and high contrast theming:

- **Guided Development** extension
- **Page Map**
- **Page Editor**
- **Application Info**

Keyboard navigation provides a streamlined experience enabling users to use the application without needing to use their mouse. Use the arrow keys to navigate within sections, the Tab key to navigate to new sections and controls, Shift + Tab to navigate back to sections and controls, and Enter to make selections.

High contrast theme provides better readability and accessibility when the theme is chosen in **Preferences** > **Color Themes** in VS Code or SAP Business Application Studio

Migration

SAP Fiori tools provide a migration utility to help move your SAP Fiori projects from other services, such as SAP WebIDE, to VS Code or SAP Business Application Studio.

Some of the key points for migration:

- Projects cloned or imported into the workspace are auto-detected for migration.
- The migration tool supports SAP Fiori elements, SAPUI5 freestyle and SAPUI5 Adaptation Projects.
- Migration tools allow the migration of multiple projects at once.
- Projects that have been migrated will not update any deployment configuration that was already defined. Please run `npm run deploy-config` after migration to configure the migrated application for deployment.
- A project must have the main data source specified in the `manifest.json` file.

Prerequisites

- Ensure that you've the latest SAP Fiori tools extensions installed. For more information, see Getting Started with SAP Fiori Tools.
- Ensure that your SAP Fiori project is running as expected in SAP Web IDE and belongs to the supported project types:

- SAP Fiori elements V2.
- SAP Fiori elements V4.
- SAPUI5 freestyle.

Recommended

- In SAP Business Application Studio, a destination of the target system needs to be defined that reflects the system used in the original application. For more information, see SAP Business Application Studio Help Portal.

  ⓘ Note

  If the target system isn't defined, previewing an application with a live data won't succeed.

- In VS Code, you need to know the target system hostname and/or client.
- We recommend that the project to be migrated is under a version control system.

Migration Steps

1. Clone your project from git by using the command line or import your project to the workspace as follows:
   - Click **File** > **Open Workspace** and create a workspace in the projects folder.
   - Export the project from SAP Web IDE to your computer.
   - Unzip the project on your computer.
   - Drag the project to the workspace of VS Code or SAP Business Application Studio.
2. Click **Start Migration** from the pop-up window appeared.

   A new **Migration** view opens listing all the projects from your workspace. For each project listed, the type of the project is also displayed. If the tool finds no suitable project for migration in your workspace, a message is shown.

   → Tip

   You can also open the Migration view anytime by starting typing `Fiori: Migrate Project for use in Fiori tools` in the **Command Palette**.

3. Select the project that you want to migrate from the list of projects by checking the corresponding checkbox. You can chose to manually add a project from the filesystem by clicking Add Project. If the supplied folder has an application that can be migrated, it's added be to your list of projects.
4. For each listed project, if applicable, fill in the respective columns based on the information provided low:

Name	Description
Application Identifier	The identifier of the project from the `manifest.json` or `pom.xml` .
Project Path	The file path to the location of the project.
SAP System	A dropdown detailing the SAP system that should be used in migration. The dropdown lists all the saved systems in VS Code or all the destinations available in SAP Business Application Studio. In SAP Business Application Studio, an entry is preselected if the destination name found in `neo-app.json` matches exactly with a destination available to the user in their subaccount. Selecting an SAP System from the dropdown sets the hostname and client automatically.

Name	Description
Destination	A free-text field that by default contains the system name from the project being migrated. It should default to the destination from the source project `neo-app.json`. Destination is only used by Fiori tools in SAP Business Application Studio and not in VS Code. To allow a project to be compatible please provide a Destination and Hostname that is accessible to both. It is recommended to use a destination for the front-end Server that has SAPUI5 libraries installed rather than connecting to the back-end **OData** server directly.
Hostname	An input box detailing the back-end hostname to be used in migration. Should be a valid **URL** or blank. This hostname is only used by SAP Fiori tools in VS Code.
SAP Client	An optional numeric field detailing the SAP client to be used in migration. Should be provided in case the client isn't the system default.
SAPUI5 Version	A dropdown detailing the version of SAPUI5 to be used when previewing the migrated application locally. Defaults to the version from `neo-app.json` if defined, otherwise uses the `minUI5Version` in the `manifest.json` file.

5. Click **Start Migration**.

   At this point, your selected projects are migrated and required packages are installed.

   > **Note**
   >
   > After the successful migration of the SAP Fiori elements project, you can view **Application Information** for the respective project. To do so, click **View Info** under the **Action** column of the migration results view. The **Application Information** tab is displayed automatically if only one SAP Fiori elements project is migrated.
   >
   > For more details, see Application Information.

Alternate Steps

To add new extensions to your extension project from SAP Web IDE, follow the steps described in this section.

Prerequisite

Copy the destination from the SAP Web IDE subaccount to the subaccount:

1. Find the name of the destination to be copied from the `neo-app.json` file of the project extension.
2. Export the destination from the Neo subaccount.

   See Exporting Solutions.
3. Go to the subaccount and import the destination.

   See Import Destinations.
4. Add the following new property:

Property	Value
`HTML5.DynamicDestination`	`true`

5. Make sure that the following properties were added:

Property	Value

Property	Value
`WebIDEEnabled`	`true`
`WebIDEUsage`	`odata_abap,dev_abap`

See Connecting to External Systems.

6. Configure the Cloud Connector to your system.

   See Cloud Connector.

Procedure

1. Clone your extension project to from your Git repository or import the project that you exported from the SAP Web IDE workspace.
2. Make sure that all the resources created in SAP Web IDE are included in this project, including the `.che` folder and the `neo-app.json` file.
3. In the popup window that appears, click **Start Migration**.

   The **Migration** view opens.

   > **Note**
   >
   > You can also open the **Migration** view by entering `Fiori: Migrate Project for use in Fiori tools` in the command palette.

4. In the **Migration** view, perform the following steps:
   a. Select the destination associated with your original application system (the destination from the section).
   b. Select an SAPUI5 version.

      Currently, the minimum version required is 1.71.0.

   c. Start the migration.

      See Migration steps for more information.
5. Verify that the `.extconfig.json` configuration file was created under the extension project's root folder, which is required for further development of this extension project in .

Development Actions after Migration

You can add extensions, preview, and deploy them. For more information, see .

Files Updated During Migration

The migration process modifies several files in your existing project. The following table lists the files to be updated during migration.

Name	Description
`.gitignore`	A new file to be added with the build artifacts and libraries ignored.
`package-lock.json`	A file deleted and re-created during migration to reflect updated libraries.

Name	Description
`package.json`	• Removes SAPUI5 Web IDE specific libraries. • Updates the npm scripts with SAP Fiori tooling targets and dependencies. • Presence of `sapux:true` and `ui5-tooling` library identifies the project as one that supports SAP Fiori tools.
`ui5-local.yaml`	A new file that supports offline development, downloads ui5 libraries locally, and runs against mock data.
`ui5.yaml`	• Removes the SAP WEBIde builder tasks. SAP Fiori deployment tasks are added later when you add deployment configuration to your project. • Reminds of file changes, supports the proxy middleware and live reload functionality.
`index.html`	A new file supporting a stand-alone preview of your application without SAP Fiori launchpad. Now, you can start your app with or without FLP.
`manifest.json`	It's populated by SAP Fiori tools. Previously, it was populated by Maven.
`changes_loader.js`	A new static file that supports live reload of the application.
`changes_preview.js`	A new static file that supports `.changes` file from adaptation updates.
`flpSandbox.html`	This file is updated by migration to support the preview and changes loader. The following differences to the old file version can be defined: • The migration tool changes the paths to the ui5 libraries so that they're proxied through the backend and allows the update of the ui5 version at runtime. • Updated to recommended config.
`locate-reuse-libs.js`	A new static file that finds any custom reuse libraries referenced in your manifest file and queries them with the back-end server to see if the libraries are installed. If so, it registers them at runtime so that they're available.
`flpSandboxMockServer.html`	Applies the same changes as the `flpSandbox.html` file but with the mock server support.

ℹ Note

If the source project does not contain a `webapp` folder, then one is created during migration and the relevant html and javascript files are moved to this location.

Verification

1. Review the file changes in the **Source Control** view. Check for any project-specific changes that may be overwritten and consider reapplying as appropriate.
2. To ensure your SAP Fiori tools application works as expected, launch any of the SAP Fiori tools commands, such as **Page Map**, **Fiori: Open Application Generator**, or **Fiori: Open Guided Development**.

ℹ Note

Some SAP Fiori apps might be missing files like `metadata.xml` before migration, which can impact some of the SAP Fiori tools features. To avoid this, after migration, make sure that you sync the `OData` service using **Service Manager** as described here: Managing Service and Annotations Files.

Related Information

Blog: Migrate SAP Fiori projects from SAP Web IDE to SAP Business Application Studio

Importing an Application

You can manually import an existing SAP Fiori application from the SAPUI5 ABAP repository to SAP Business Application Studio or VS Code.

Preparation Steps

Before importing an application, in SAP Business Application Studio or VS Code, in your workspace, create new folders with the following names:

Name	Description
`restore-from-exported`	This folder contains the restored app.
`restore-from-exported/webapp`	This folder will contain the content of the downloaded zip/tgz.

Import Steps

To import SAP Fiori apps from the SAPUI5 ABAP repository, perform the following steps:

1. Login to your SAPUI5 ABAP backend system and navigate to the transaction `SE80`.
2. Run the report `/UI5/UI5_REPOSITORY_LOAD`.
3. Provide the name of the SAPUI5 application and click Download.
4. Choose an empty folder for the download target.
5. From the resulting view, download all as a zip using the button Click here to Download at the end of the page.
6. Extract the zip into the folder `restore-from-exported/webapp`: Verify that the `manifest.json` is located at `restore-from-exported/webapp/manifest.json`.
7. Create a package.json in the folder `restore-from-exported`. It should contain the following where the value for `name` matches the application name in the `manifest.json`.

   ```
   {
      "name": "sap.fe.demo.awesomeapp",
   }
   ```

   You can see the original application name and namespace in the `manifest.json` file. For example:

   ```
   {
   "sap.app": {
   ```

```
"id": "sap.fe.demo.awesomeapp",
..
}
```

8. In SAP Business Application Studio or VS Code workspace start the Migration command `Fiori: Migrate Project for use in Fiori tools` if not already prompted to do so.
9. The projext should be found in `restore-from-exported` and listed for migration.
10. Choose the appropriate options and migrate the project.
11. Upon successful completion the project should now be compatible withSAP Fiori tools.

Telemetry

You can enable/disable collection of usage analytics data for SAP Fiori tools. If enabled, non-personally identifiable information is used to help understand the product usage and improve the tool.

To enable/disable telemetry:

- In VS Code or SAP Business Application Studio, execute command **Fiori: Change Telemetry Settings**.

Report Issues and Security

- See **SAP Fiori tools FAQs** to get up-to-date information.
- Check the SAP Fiori tools Community.
- If you can't find an answer in **SAP Community** or need additional assistance, create an incident in **SAP Support Portal** under component: **CA-UX-IDE**. See Contact SAP Support

Security

The following best practices are suggested when usingSAP Fiori tools: :

- Follow the software development security guidance provided by your organization.
- Avoid using **OData service** hosted on a production system for developing SAP Fiori application.
- Don't use the same user credentials to access development and production systems for the OData service.
- Use a trusted NPM registry.
- Follow the security guideline for the prerequisite software where such is provided.
- Ensure that you use a source control system and regularly committing code to it.
- See Security about additional information about security.

Related Information

SAP Fiori tools FAQ
SAP Fiori tools Community
Contact SAP Support

Project Functions

The following project functions are available when using SAP Fiori tools:

- Application Information
- Reuse Library Support
- Data Editor
- Delete an Application in CAP Project
- Environment Check
- Managing System Connection
- Managing Service and Annotations Files
- Project Validation
- Viewing Service Metadata

Application Information

When your SAP Fiori elements project is generated, an **Application Information** page will launch automatically. The page consists of four sections:

- **Project Detail**. Provides information about the project, such as project type, SAPUI5 version, backend, and pages that are part of the application. Click the page icon to launch the Configure Page Elements.
- **Application Status**. A summary of the project dependencies with links to fix any issues.
- **What you can do**. Quick links to Fiori tool commands that are relevant to the project.
- **What you can learn**. Links to various help topics including how to contact support.

You can relaunch the **Application Information** page at any time by executing the `Fiori: Open Application Info` command. In addition, you can also run Project Validation at any time after generating your project.

Application Minimum SAPUI5 Version

The minimum SAPUI5 version is declaring the version which is required at runtime to support all the features used in application development. If the target system for deployment doesn't have the required minimum, the deploy application of SAP Fiori tools helps with a warning. The version must be defined by the developer. This can be done during generation when selecting a version for the project. Later changes can be done with the command *Fiori: Change Minimum SAPUI5 Version*.

The application's minimum version is also used to determine which corresponding version of @sap/ux-specification is installed along the project to provide the matching feature set in application modeler and Guided Development.

Reuse Library Support

Currently, a library to be used in your project has to reside in your workspace, in either SAP Business Application Studio or VS Code. With SAP Fiori tools, you can reuse a library or component in your SAP Fiori apps by adding a reference to another project.

Adding Reference to Reuse a Library

Prerequisites

1. A reuse library project is cloned or imported into your workspace.
2. An SAP Fiori project is already presented in your workspace.

Steps

To add a reference for reusing an SAP Fiori library, perform the following steps:

1. From the command palette, execute the command **Fiori: Add Reference to SAP Fiori Reusable Libraries**.
2. From the **Project Folder Path** drop-down list, select the SAP Fiori project that exists in your current workspace.
3. Select **Reusable Library Source** as a workspace.
4. Select reusable libraries or components from the list.
5. Click Finish.

When you click Finish, the following files get updated with a reference to the selected reuse library:

- `ui5.yaml`
- `ui5-local.yaml`
- `manifest.json`

Deploying a Reuse Library Project using SAP Fiori tools

Deploying a Reuse Library Project using SAP Fiori tools.

1. Add a Reuse Library project to your workspace
2. In the resulting migration pop up window, click on Start Migration
3. From the list, select the project which you would like to migrate
4. Now, click on Start Migration
5. Once the dependencies are installed, you are ready to use SAP Fiori tools to deploy to an ABAP environment.
6. To generate deployment configuration, please follow the instructions here Generate Deployment Configuration ABAP
7. To deploy the project to ABAP environment, please follow the instructions here: Deployment to ABAP

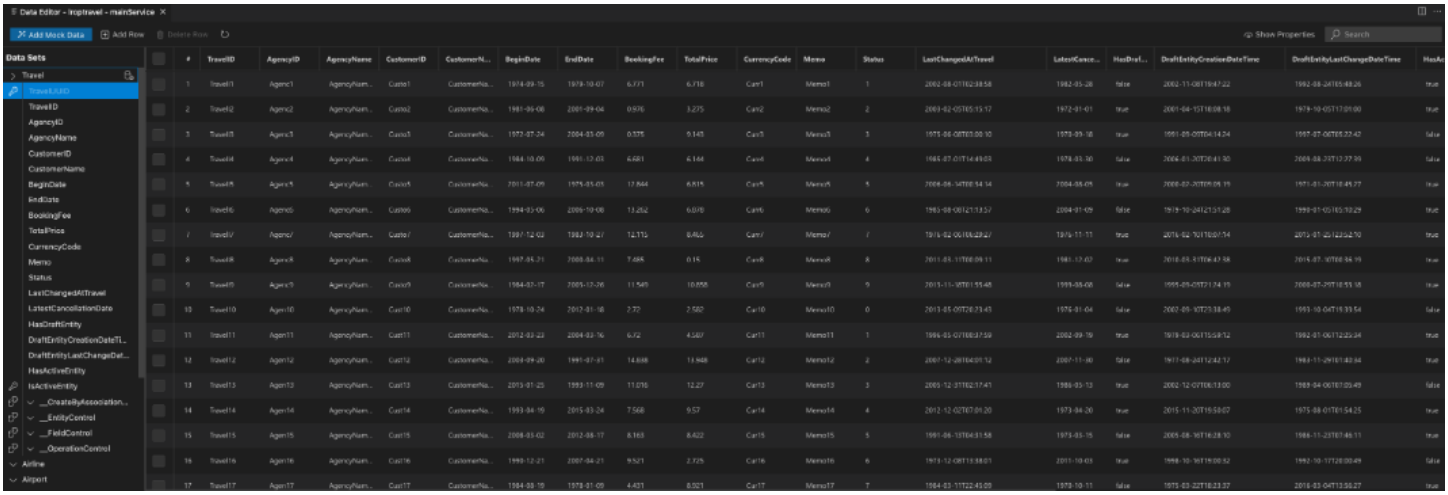
Data Editor

Previewing the application using `npm run start-mock` generates mock data on the fly. If you want to generate mock data and store it in the `.json` file format, you can right-click on your project and launch `Open Data Editor`. Once generated, mock data is stored in the `.json` format under the `/webapp/localService/mockdata` file.

ℹ Note

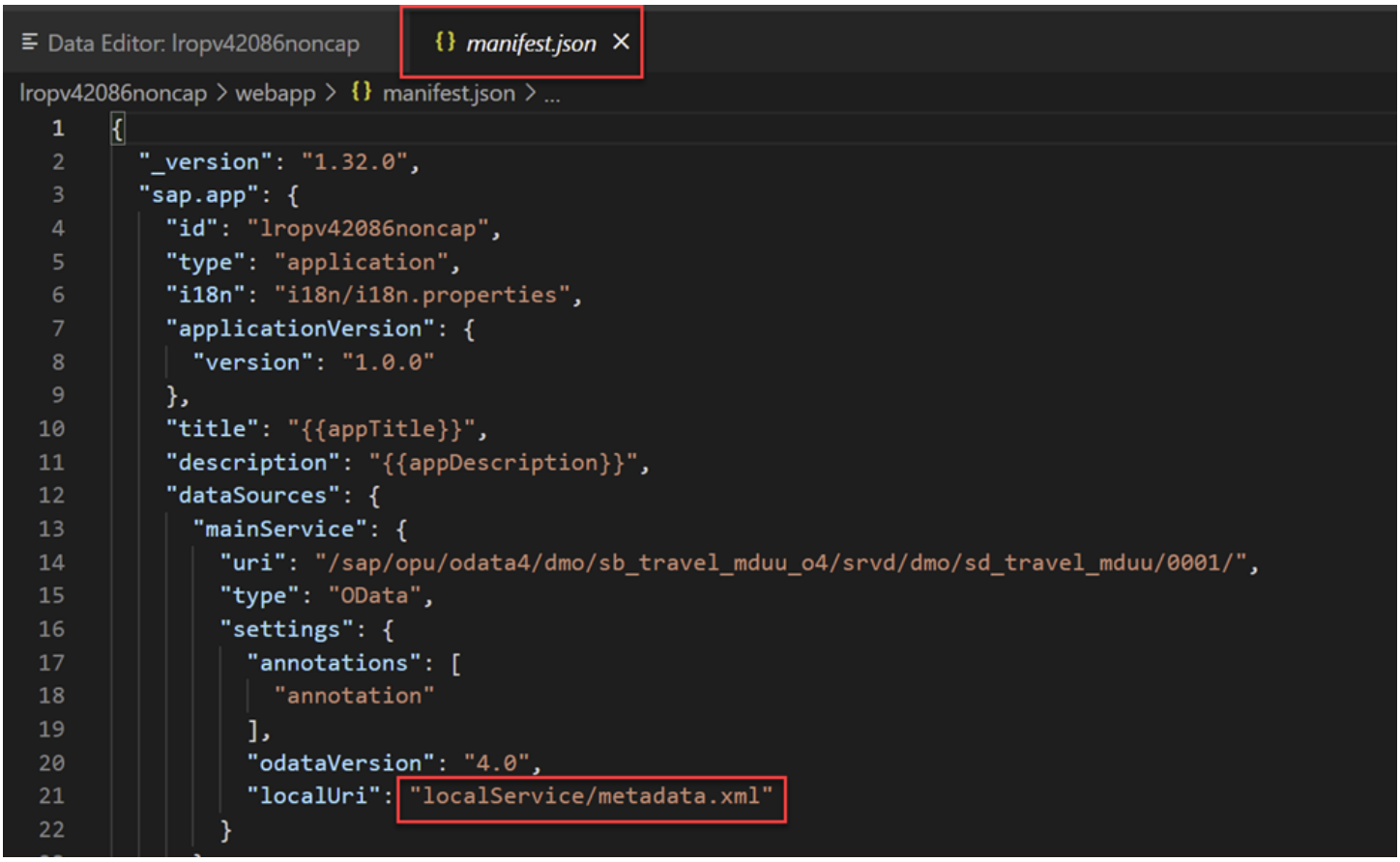
Mockserver configuration is needed prior to using `npm run start-mock`. See Installing MockServer.

For templates like **Overview page** where there are multiple services, you will see a drop-down list to select the metadata file you want use. If your application is running, you can stop the mock server by pressing *Ctrl+C*.



Data Editor reads the `metadata.xml` file that is defined in the `manifest.json` under the **dataSource** and generates mock data based on the property type.

Data can be edited by either double-clicking in the cell of the Data Editor or by editing the `Entity.json` file.

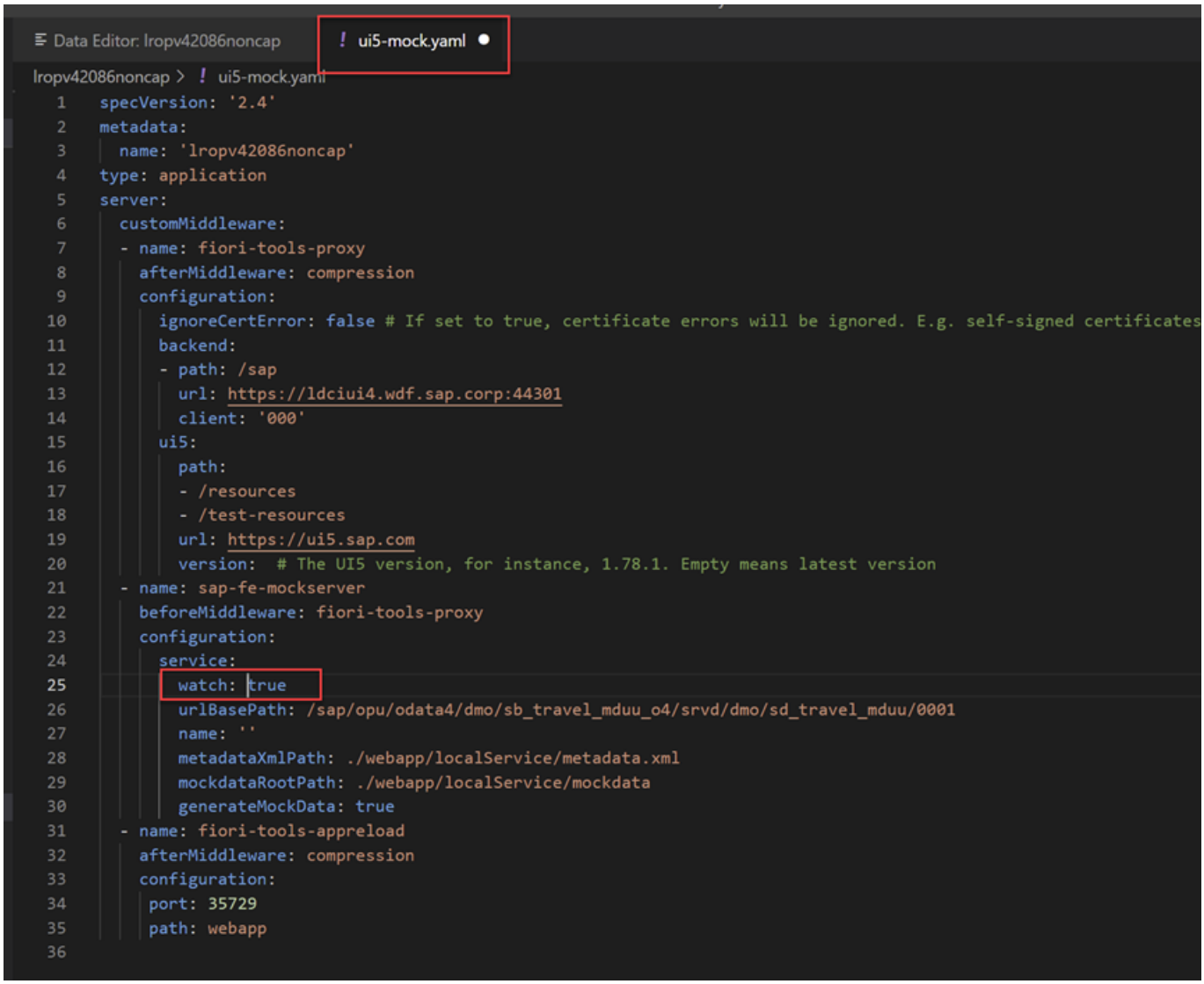


If the **mock server isn't running** you can:

- Make changes in the Data Editor, automatically reflected in the `.json` file.
- If changes are made in the `.json` file, hit the refresh button to update the canvas.

If the mock server is running:

- Add **watch: true parameter** to the `ui5-mock.yaml` file in order to pick up the changes.



Editing Mock Data

Editing Data - **double-click** in the editable cell.

ⓘ Note

Primary keys and foreign keys aren't editable.

Add Row - add one row at a time to your data by pressing the Add Row button.

ⓘ Note

Additional rows will be added automatically to all the entities that are associated via a foreign key.

Delete Row - select any row or rows and press the Delete Row button.

ⓘ Note

Rows will be automatically deleted on all the entities that are associated via a foreign key.

Searching Mock Data

To search for mock data, perform the following steps:

1. Click on the Search input field in the header bar of the Data Editor .
2. Enter search criteria in the search input field.
3. Select the mock data that matches the search criteria in the drop-down table. The data selected is highlighted in the Data Editor table.

Show and Hide Properties

To show properties in the Data Editor table, perform the following steps:

1. Click on Show Properties in the header bar of the Data Editor to open the popup.
2. Check the properties you want to display in the Data Editor table.
3. Click Save.

To hide properties in the Data Editor table, perform the following steps:

1. Click on Show Properties in the header bar of the Data Editor to open the popup.
2. Deselect the property that you don't want to display in the Data Editor table.
3. Click Save.

ℹ Note

Certain property groups are hidden by default and can be shown using the Show Properties functionality. You can also search for a particular property using the search input field in the popup.

Delete an Application in CAP Project

ℹ Note

For more information about CAP services, see: https://cap.cloud.sap/docs/about/.

You can safely delete an SAP Fiori application created inside CAP projects with the SAP Fiori elements or SAPUI5 freestyle generators. During the deletion process, the folder containing the application is deleted along with all changes in global files, such as references to already invalid annotations in a subfolder.

Perform the following steps:

1. By using **Command Palette** in VS Code (CMD / CTRL + Shift + P), execute the **Fiori: Delete Application from CAP project** command.
2. Select the application to be deleted from the dropdown list of applications available.
3. A dialog box **Do you really want to delete application <application_name>?** with the following options appears:
   - **Yes**
   - **No**
   - **Cancel**
4. Select **Yes** to delete the required application and revert all changes in global files.

ℹ Note

If you select either **No** or **Cancel**, the application will remain the same without any changes applied.

Environment Check

Destination Checks: SAP Business Application Studio

SAP Fiori tools environment check creates a report that the users can use to identify and change the issues. In addition, the report also contains information that can be very useful for SAP Product Support to gather the initial set of information to further process an incident. Please note that SAP Fiori tools environment check is a tool to help troubleshoot some common destination-related issues. To use SAP Fiori toolsenvironment check:

1. In SAP Business Application Studio, execute the command `Fiori: Open Environment Check` and choose `Check destination`.
2. Choose a `destination` from the quick pick.
3. If prompted, enter credentials.
4. To view the results, choose View the results.
5. To create a zip file to be shared with SAP Product Support, choose View and Save results.

The environment check report for SAP Fiori tools has the following sections:

- **Environment**: It will display all the important details like Dev Space type, node version, etc. of the environment.
- **Destination Details**: It will display chosen destination-specific information like destination parameters. It may also contain some messages specific to SAP Fiori tools usage.
- **All Destination Details**: This section will list all the destinations and its properties available to the user in the current subaccount.
- **Messages**: This section will have raw log messages of chosen destination which can be useful for SAP Product Support.

Gather Environment Information: SAP Business Application and VSCode

SAP Fiori tools environment check can also check and create report on the development environment in SAP Business Application Studio or VS Code. This information will be very useful to add to a support ticket so that SAP Product Support has relevant information about your development environment to start with. For e.g. You can have an older version of SAP Fiori application generator, some required npm package missing, etc..

To get the environment information, follow the steps:

1. In SAP Business Application Studio or VS Code, execute the command *Fiori: Open Environment Check* and choose `Gather environment information`.
2. To see the results immediately, choose View and Copy results. The environment check details will also be copied to your clipboard for easy access.
3. To see the results and create a zip file to be shared with SAP Product Support, choose View and Save results. You see a report that has details like tools, and npm packages installed along with the version numbers.

Managing System Connection

SAP Fiori tools running in VS Code enable to save the connection information to a remote system. This functionality provides faster authentication while creating an application, generating deployment configuration, and deploying an application. The credentials are saved in the operating system secured storage, such as Credential Manager in Windows and Keychain in Mac.

View Saved Systems Details

To view the saved systems details, perform the following steps:

1. On the activity toolbar from the left side, click **SAP Fiori** (the wrench and pencil icon).
2. Expand the **SAP Systems** view.

   You can see the list of saved systems along with the usernames used for authentication.
3. To see the stored system details in a new view, click a particular system entry or right-click and select.**Show SAP System Details**

Test Saved Systems Connection

To test the connection of a saved system, perform the following steps:

1. Open the details of a saved system. For more information, see View Saved Systems Details.
2. Click **Test Connection**.

   As a result, you'll see if the system connection was successful and whether it supports OData V2 and/or OData V4 services.

ⓘ Note

We recommend that you test a saved system connection to ensure the service catalog of a selected system works as expected.

Edit Saved Systems

For editing an existing SAP system connection detail, perform the following steps:

1. Right-click a saved system name and click `Show SAP System Details` button or click a saved system name to open system details tab to open SAP `system Details`
2. For ABAP `On Premise`, update any of the following fields.
   - System Name
   - URL
   - Client
   - Username
   - Password
3. For ABAP `Environment on SAP Business Technology Platform`, update the following fields:
   - System Name - editable
   - URL - noneditable (determined)
   - Client - non-editable (determined)
   - Service Key - editable
4. Click on `Test Connection`
5. Click `Save`

ⓘ Note

It is recommended for you to test the connection before saving to ensure it is working as expected.

Delete Saved Systems

To delete the saved systems, perform the following steps:

1. On the activity toolbar from the left side, click **SAP Fiori** (the wrench and pencil icon).
2. Expand the **SAP Systems** view.
3. Select any saved system.
4. Click the **Delete** icon next to the system name.
5. Click **Yes** in the confirmation dialogue box.

Create New System

To create a new ABAP On Premise system, perform the following steps:

1. Click + icon with tooltip **Add SAP System** on the right side of system panel.
2. Enter valid values for ABAP On Premise system that you have access to:
    - **System Name** - your choice
    - **URL** - system URL
    - **Client** - usually 3 digits, leave empty if not required
    - **Username** - your username
    - **Password** - your password
3. Click Test Connection, verify that message is shown: **This SAP system connected successfully**.
4. Click Save.
    - Message **System information saved**is displayed.
    - Saved system is shown in systems panel in a format: **<name of added system> [username]**.

ℹ Note

You can still save the new system without first validating that the connection succeeds. However, it is recommended that you test the connection before saving to ensure it is working as expected.

To create a new **ABAP Environment with service key**, perform the following steps:

1. Click + icon with tooltip **Add SAP System** on the right side of system panel.
2. Click System Type, select **ABAP Environment on SAP Business Technology Platform** and enter valid values for **ABAP Environment on SAP Business Technology Platform** system that you have access to:
    - **System Name** - your choice
    - **Service Key** - copy/paste service key of your chosen SAP BTP system
3. Click Test Connection - authentication page will open in browser. If required, enter your SAP BTP system user email/password to log on via browser. After successful authentication go back to VS Code and verify that:

- URL value is filled in by system
- message is shown: **This SAP system connected successfully.**

4. Click Save:
   - Message is shown: **System information saved.**
   - Added system is shown under systems panel in a format **<name of added system> (BTP)**.

Export/Import an existing ABAP on Premise SAP System

To export an existing ABAP On Premise system, perform the following steps:

1. Right-click a saved system name and click Show SAP System Details or click a saved system name to open **SAP Systems** details.
2. Click Export System. A copy of the saved system will be downloaded in JSON format. Please note that no sensitive credential information is included in the exported JSON file.

To import an ABAP On Premise system, perform the following steps:

1. On the activity toolbar from the left side, click SAP Fiori,  .
2. Alongside the SAP Systems title bar, click on the Import SAP System,  .
3. Navigate to the the JSON file that you would like to import.
4. Upon successful import, provide your credentials for that system and click Test Connection.
5. Once the connection is successfully tested, click Save to finish importing the system.

ℹ Note

If you already have an SAP Saved system locally with the same name, you will asked to confirm before overwriting.

Managing Service and Annotations Files

Manage Service Files

How to add services to a project

ℹ Note

The ability to add services is only supported for the SAP Fiori elements Overview Page and SAPUI5 freestyle projects.

1. Right-click the application project `manifest.json`.
2. In the context menu, select **Service Manager**.
3. Click **Add Service**.
4. Choose Connection Type:

a. Destination (SAP Business Application Studio) - Select server destination from the dropdown. Enter username and password if needed.

b. SAP System (VS Code) - Select server SAP System from the dropdown. Enter username and password if needed.

c. Hostname - Enter the server hostname, SAP Client, and username and password if needed.

5. To specify the OData service URL

   a. Enter Service URL manually.

   b. Fetch Services from the server catalog and select from the dropdown list.

6. Click **Add**.

   A new service appears in a service list. A service `metadata.xml` is now added to a local service folder of the project along with the service backend annotations (if they're available).

How to Refresh a Service from the Server

1. Right-click the SAP Fiori elements application project `manifest.json`.
2. In the context menu, select **Service Manager**.
3. Click the **Pencil** icon opposite the service.
4. Choose Connection Type:

   a. Destination (SAP Business Application Studio) - Select server destination from the dropdown. Enter username and password if needed.

   b. SAP System (VS Code) - Select server SAP System from the dropdown. Enter username and password if needed.

   c. Hostname - Enter the server hostname, SAP Client, and username and password if needed.

5. Click

   - **Refresh** - Refresh local copy of metadata and annotation files of service.
   - **Refresh & Save** - Refresh local copy of metadata and annotation files of service and save connection to UI5 yaml files.

How to Delete a Service

1. Right-click the SAP Fiori elements application project `manifest.json`.
2. In the context menu, select **Service Manager**.
3. Click **Delete** icon.

The `metadata.xml`, related annotation xml files and mockdata is deleted from the project. Also, the `ui5*.yaml` files will removed any backend routing and mockserver entries specific to the service being deleted.

For more information about mockserver, see Use Mock Data.

Manage OData Annotations Files

The OData services can have multiple local annotation files associated with a service. The Annotation File Manager can be used to manage local annotation files associated with a service.

ℹ Note

Managing annotations is limited to the OData service and not applicable to CAP CDS.

How to open the Annotation File Manager

1. Right-click the `manifest.json` file in the SAP Fiori elements application project.
2. In the context menu, select **Annotation File Manager**.
3. Select the target service from the drop-down list.

Alternatively, you can click **Annotation Hierarchy** in the **Annotation List View** for a particular projection or property of service.

How to add annotation files in the Annotation File Manager

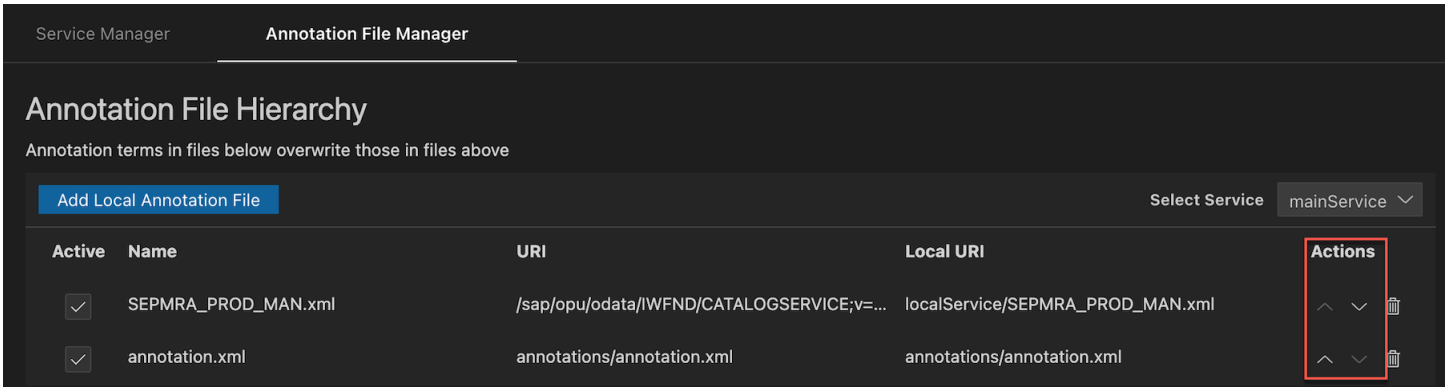
1. Right-click the `manifest.json` file in the SAP Fiori elements application project.
2. In the context menu, select **Annotation File Manager**.
3. Select the target service from the drop-down list.
4. Click **Create Local Annotation File**.
5. Fill in the criteria required for creating an annotation file.
6. Click **Create**.

The newly created annotation file appears in the **Annotation File Manager** for that service, and also in the **Annotation List View** for the target projection.

How to change the hierarchy of local annotation file

> **Note**
>
> The highest-ranked annotation file in the **Annotation File Manager** table is at the bottom resembling precendece rules in the `manifest.json`.

1. In the **Annotation File Hierarchy** use the up/down arrows to change the hierarchy.



How to activate and deactivate local annotation files

1. Right-click the `manifest.json` file and select **Open Annotation File Manager**.
2. Select or clear the checkbox in the active column of the **Annotation File Manager** table to activate or deactivate the annotation.

> **Note**
>
> When a file is deactivated, it's no longer considered a part of the annotation file hierarchy.

How to delete an annotation file

1. Select the active checkbox.
2. Click the delete icon next to the annotation file.

Project Validation

When the project is generated, you can validate it by executing `Fiori: Validate Project`. Use a quick pick selection to select a project for validation from the multiple projects in the workspace.

The validation comprises of the following steps:

- **project**

  In the project step, a validation checks existence and semantic correctness of the mandatory files, such as `package.json`, `manifest.json`, and `ui5.yaml`. In this step, these files are read and mandatory entries are checked, such as the existence of `devDependencies` in `package.json`.

- **annotation**

  The annotation step validates the project annotation files using the same modules as uses the extension SAP Fiori tools - XML Annotation Language Server. It reports the same results as if all local and backend annotation files are opened in the project one by one.

- **specification**

  In the specification step, the content of `manifest.json` and change files in the changes folder are validated in detail. The module `@sap/ux-specification` is used to **import** the project and in return gives messages in case of invalid configuration.

- **eslint**

  The validation step for a projects checks if `eslint` is installed as a dependency. If it is, the project validation runs the `eslint` check based on the project's configuration and the rules defines in the `eslint-plugin-fiori-custom`. The results of this check are then collected. You can enable `eslint` support for a new project by selecting the appropriate options in the Advanced Options of the SAP Fiori application generator, as described in the Additional Configuration section.

After all validation steps are executed, a report is written and displayed as a markdown (.md) file and in the **Problems** tab of VS Code or SAP Business Application Studio.

Viewing Service Metadata

The Service Modeler supports the visualization of the OData V2/V4 Service Model based on the `.xml/.edmx` files and CAP CDS services.

With the SAP Fiori tools - Service Modeler extension, you can perform the following operations::

- Easily browse complex services and view the entities, projections, properties, and associations of a service.
- View annotations associated with the entities, projections, and properties of a service.
- Manage services required for SAP Fiori elements applications, such as adding new services to a project or syncing services that already exist to ensure they are up to date.
- Manage local annotation files associated to the different OData services of a SAP Fiori elements project.

Launch Service Modeler

Service Modeler can be launched in several ways.

Use Command Palette.

- Open Command Palette.
- Start typing ***Service Modeler***.
- Select SAP Fiori tools: Service Modeler: Open Service Modeler.
- Select SAP Fiori elements project from your workspace.

> **Note**
> If the project contains multiple services, you need to specify the service that you want to visualize.

Use folder context menu.

If you already have an SAP Fiori elements project in your current workspace, you can right-click any folder within your project and Open Service Modeler.

From the Text Editor.

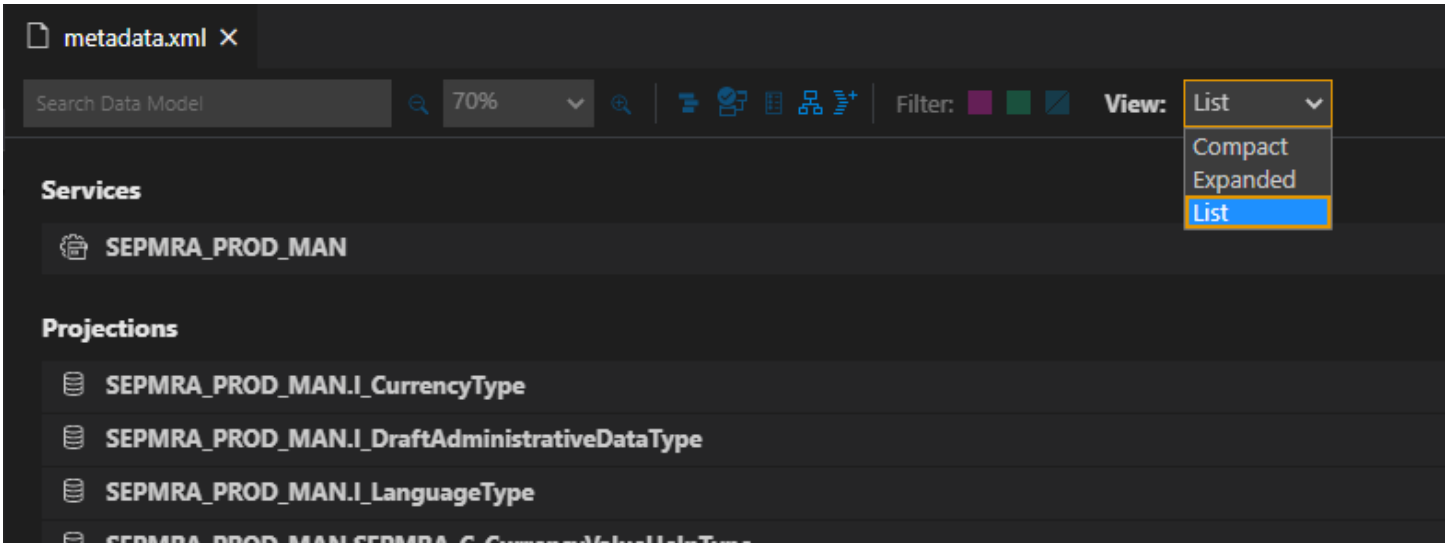
If your `metadata.xml file` is open in the text editor, click the annotations icon .

Use Service Modeler for a Service

Visualize a Service

The service can be visualized in three different views:

- Expanded
- Compact
- List

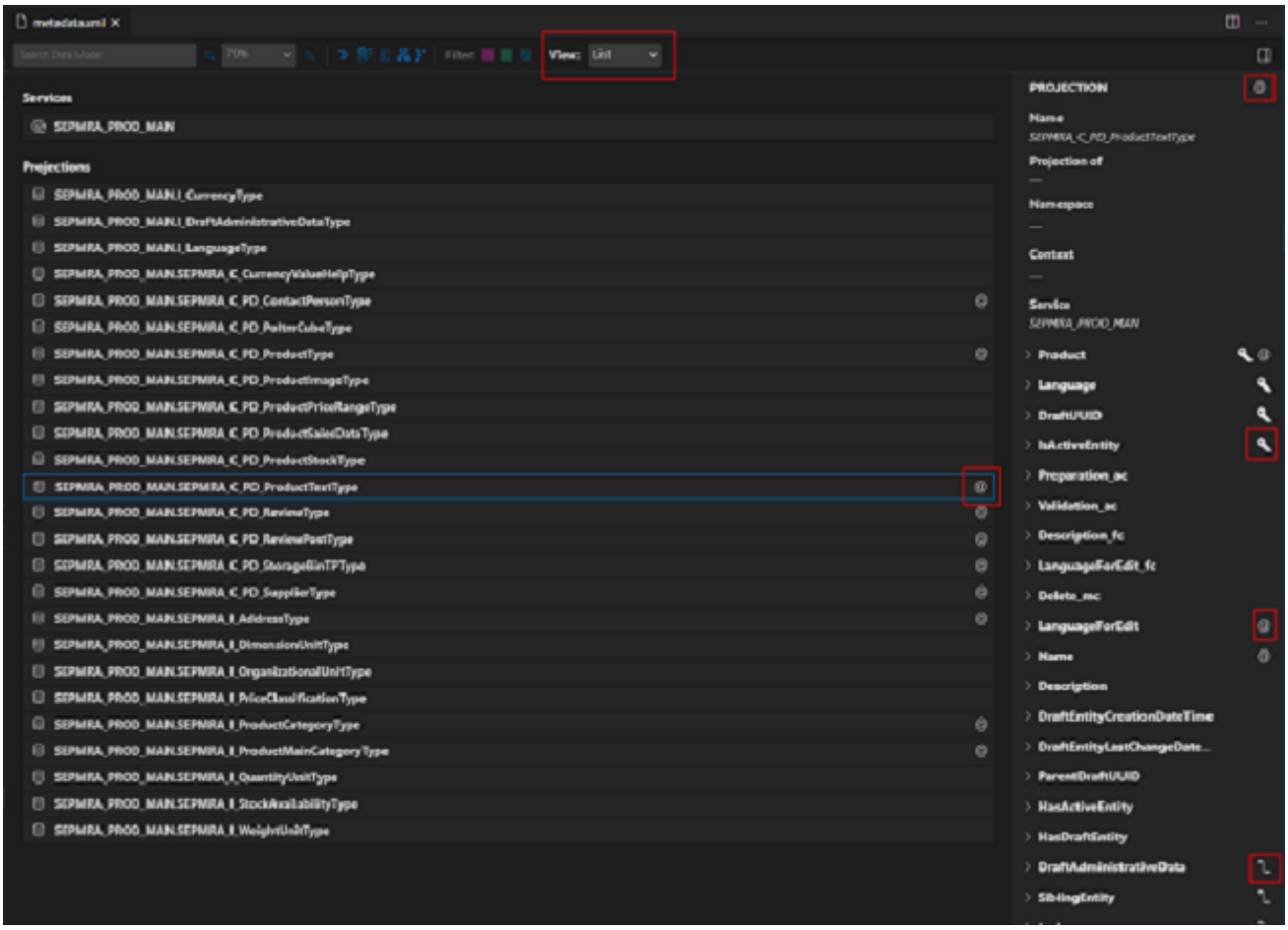


The List View is the default view on launching the Service Modeler.

List View

The List View displays the entities and projections of a service in a list format. Details of a projection or entity are displayed in the detail panel on selection. The detail panel lists the properties of the entity or projection. The details of the properties are also displayed, such as the properties name and type.

- Properties that are the primary key of the entity or projection are identified by a key icon.
- Associations are identified by a link icon.
- Properties that have annotations associated to them are identified by an annotation icon.



Expanded and Compact View

Expanded and Compact View show a graphical representation of a service. Each entity or projection in a service is represented by a node on the canvas.

- **Service Node** has a blue header. In the Expanded View, the service node lists all the entities included in this service.
- **Entities** are connected to the service nodes with a pink link.
- **Associations** between **entities** or **projections** are represented by blue links. The blue links are displayed on selection of an **entity** or projections in the service model. They can also be set by default using the filter on the service model toolbar. The association links are used between the source properties and the target properties.

  #### ℹ Note

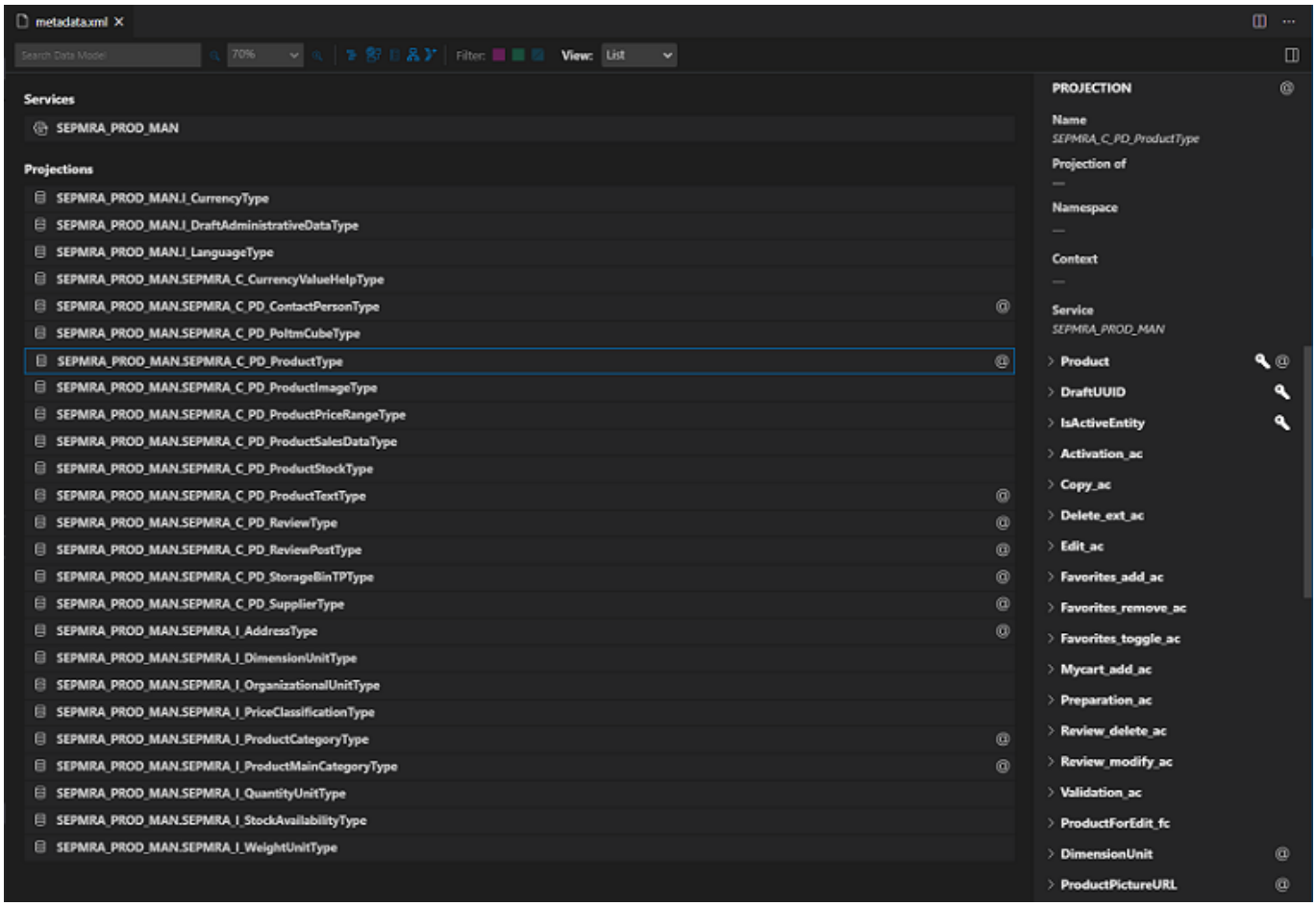
  Only one to one associations are currently supported.

- Properties of a projection or entity are displayed in the entity or projection node.
- **Properties** that are primary keys of the entity or projection are identified by a key icon.
- **Properties** that have annotations associated with the property field are identified by an annotation icon.

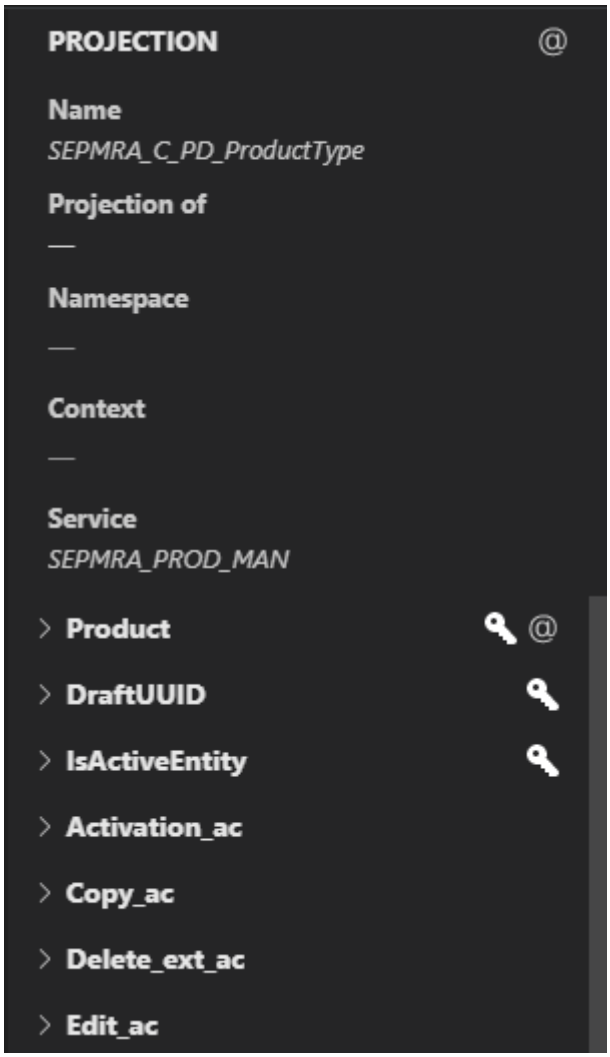
ℹ Note

In the **Compact View**, properties of the projection or entity are not displayed. As some service models can be large, the canvas may become overloaded.

- **Detail Panel**. The detail panel lists the properties of the entity or projection. The details of the properties are also displayed. Details like the name and type. Properties that are the primary key of the entity or projection are identified by a key icon .



- Associations are identified by a link icon  and properties that have annotations associated with them are identified by an annotation icon .

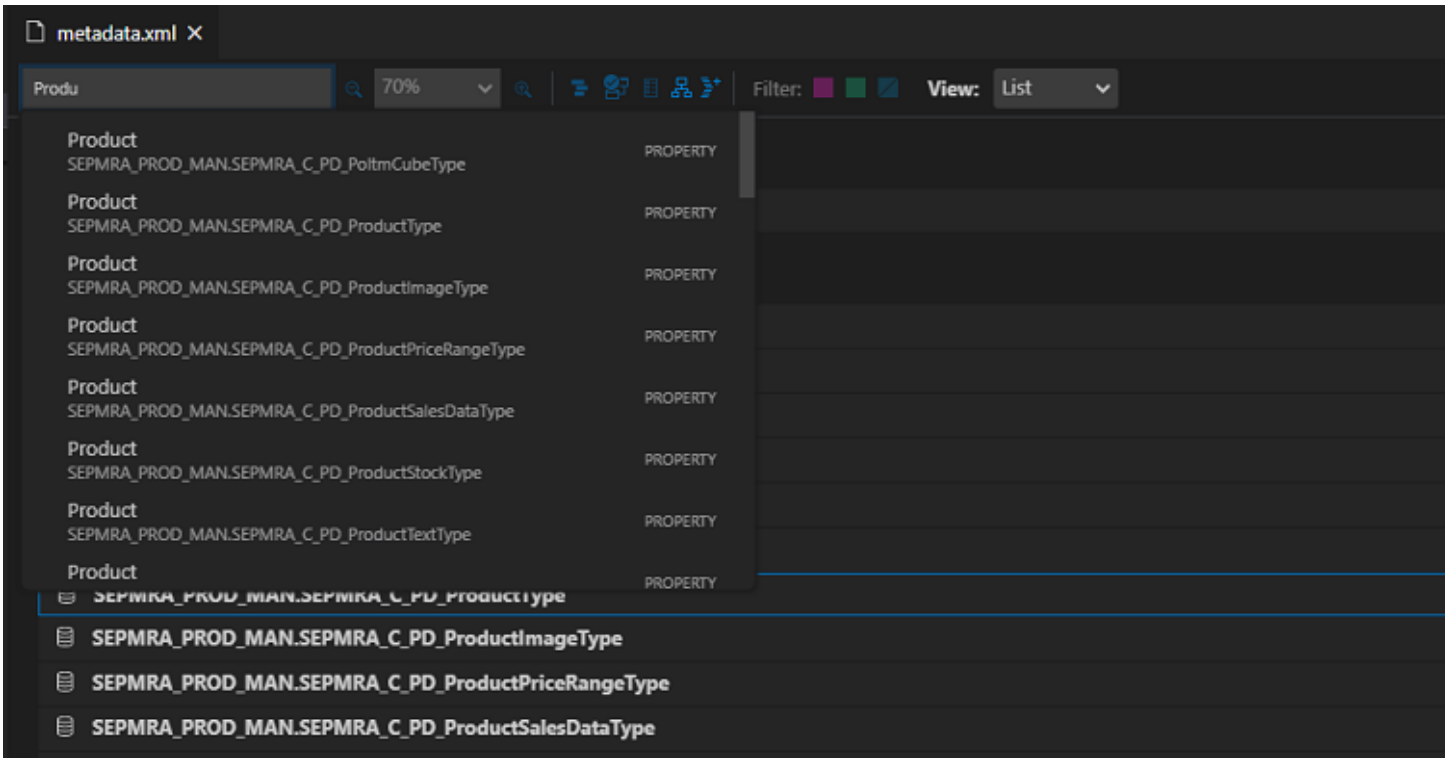


Searching for Entities

Search functionality is available in the Service Modeler toolbar.

When you enter a phrase in the search input field, results matching the search criteria are displayed in a list. The list contains the type of element that matches the search criteria. For example, an entity, a projection, or a property.

On selection of an item in the search result list, the entity or projection is selected in the view. If the selection in the list is a property, the entity or projection is selected in the view but the property in the detail panel is opened by default and scrolled into view.



How to Browse the Service

To Switch Views

The Service Modeler can be viewed graphically or in a list. To change the views, select the preferred view from the view drop-down list in the toolbar.

Zoom

Zoom functionality can be used to manage large data models. Some data models can be too large to be viewed clearly on the canvas. The zoom functionality enables users to focus on particular areas or entities of a Service Model. You can use the plus, and minus icons in the toolbar to set the preferred zoom percentage or select the preferred zoom from the dropdown list.

Redistributing and Saving Graph

When viewing a Service Modeler in the Expanded or Compact View, users can configure the layout of the **Service Model**. Users can move the entity or projection nodes around the Service Modeler canvas to make the model more manageable or readable. All association, projections, or service links adapt to the entities new position.

To move entities or projections on the canvas.

- Drag the entity to desired position, and then click Save Graph  icon on the toolbar.

ℹ Note

The updated node position is not reflected in the different views. The node position has to be updated in every view.

To redistribute the node positions of the model to its original state, click Redistribute Graph  icon on the toolbar.

Filter connection links

As some services can have multiple entities or projections, filtering in the **Expanded** or **Compact View** can help to reduce the number of connection links. To do so, use the filter to identify which connection links can be visible to make the service model more manageable and readable. To filter connection links, click the link color that can be removed from the Service Model on the toolbar. Another way to filter is to click **Show/Hide Legend**  icon and check the connection links to be displayed in the Service Model.

How to set default preferences for the Service Modeler

The Service Modeler preferred default view to display service links in **EDMX** files can be set with the extension settings of the tool.

1. Click **Extensions tab**  icon.
2. Click **settings**  in the Service Modeler tool.
3. Set default settings.

Generate an Application

To create an application in VS Code or SAP Business Application Studio, perform the following steps:

1. Using **Command Palette** in VS Code (CMD/CTRL + Shift + P), select *Fiori: Open Application Generator* or **Start from template** in SAP Business Application Studio.

   Note

   In SAP Business Application Studio, use one of the following dev spaces, depending on your development needs:

   - SAP Fiori
   - Full-Stack Application Using Productivity Tools
   - Full Stack Cloud Application

2. The **Template Wizard** appears.

   Note

   SAP Fiori application increases development productivity and ensures consistency across your SAP apps by using standard page types to build applications. This generator follows SAP Fiori elements or SAPUI5 freestyle approach to build an SAPUI5 application.

3. On the **Template Selection** page, select one of the following options from the **Template Type** dropdown list:

   - SAP Fiori (default value)
   - Deprecated templates (includes SAPUI5 templates that have been deprecated)

4. Click Next.

   Based on your selection, the next page appears.

ℹ Note

After the SAP Fiori project is generated, the **Application Information** page will open. You can relaunch the **Application Information** page at any time by executing `Fiori: Open Application Info` command.

For MTA deployment, see Generate an MTA Deployment File

Generate an Application with SAP Business Application Studio Service Center

In the SAP Business Application Studio Service Center, you can explore services from various service providers.

The Service Center displays the services from subaccount destinations and SAP API Business Hub. The exposed services can be used as a data source for creating an SAP Fiori application.

For more information, see Service Center.

Project

Once a project is generated, you can see its structure:

- `webapp`. Root folder for SAPUI5 based web applications.
- `.npmrc`. Lists any npm registry configuration updates required for the generated project.
- `.gitignore`. Specifies files or folder patterns that should be excluded from source control, such as node_modules.
- `package-lock.json`. Ensures that the project node dependencies are fixed to the correct versions; helps improve the speed of the node libraries installation routine.
- `package.json`. The main configuration file for node-based projects.
- `README.md`. The readme file providing details on the options chosen to generate the application.
- `ui5-local.yaml` . Supports local development of an application in preview mode.
- `ui5.yaml`. Ensures that the application generated locally can connect to the supplied data source and supports dynamically updating the version of SAPUI5.
- `node_modules`. Contains node modules required to install and run the current project locally (npm install creates and updates the folder). This folder doesn't need to be version-controlled.

ℹ Note

The generated project structure differs if you select the option "Use a Local CAP Node.js Project" during its generation. In this case, the preview and deployment functionality are provided by the local CAP project. As a result, the `package.json` file has a more basic content and some files aren't generated, such as `ui5.yaml`, `.npmrc`, and more. Additionally, instead of local `annotation.xml`, an `annotation.cds` file is generated so that annotations are defined in CAP CDS syntax instead of OData EDMX.

For more information about npm, see, @sap/generator-fiori-elements and @sap/ux-ui5-tooling .

ℹ Note

The SAP Fiori application generator now consumes open source libraries for writing the Fiori elements and Fiori freestyle applications. Please see Fiori elements writer and Fiori freestyle writer documentation for more information on these libraries.

Note

In SAP Fiori application generator, when some of the most common errors are shown during project generations, the generator gives an option to end user to launch Guided Answers Extension by SAP , and troubleshoot the error based on the solution provided by SAP experts.

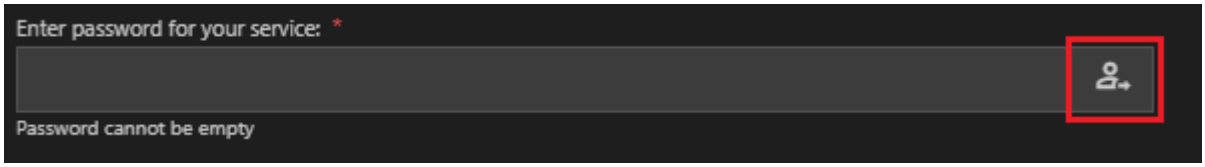
SAP Fiori Elements

In SAP Fiori elements, perform the following steps:

1. Select a **Floorplan** for your application and click **Next**. The following options are available:
    - List Report Object Page
    - Worklist
    - Analytical List Page
    - Overview Page
2. Select a **Data Source** and click **Next**.
    - Connect to an SAP System
    - Connect to an OData Service with a customized URL
    - Connect to an OData service
    - Upload a Metadata Document
    - Connect to SAP API Business Hub
    - Use a Local CAP Project

    ### Note

    If a username and password are required, enter your credentials and click **Login**.

3. Select **Associated floorplan properties**, such as the **Main entity** and **Navigation entity**.
4. On the **Project Attributes** wizard page, configure the main project attributes.

    ### Note

    Some fields are already prefilled with default text, which can be modified if needed. Mandatory fields are marked with an asterisk (*).

    - **Module name** (required). Must be alpha-numeric and **cannot contain spaces**. The generated NodeJS application uses the module name as its package name. It is used as the folder name of the generated application.

        ### Note

        Module names can only contain URL-friendly characters.

    - **Application title**. The title that is displayed in the launchpad tile and header of the generated application.

- **Application namespace**. The SAPUI5 project namespace to be used. Must start with a letter and contain letters, digits, and periods only.
- **Description**. The description of the application.
- **Project folder path** (required). The parent folder in which the new application is generated. The new application is generated in a new folder with the module name. If a folder with the same name already exists, the user must choose a new module name.
- **Minimum SAPUI5 version**. From the drop-down list, select the minimum SAPUI5 version that the application will require.
  - The dropdown will show the list of available versions of SAPUI5, with the current **default** version being pre-selected. The dropdown will list SAPUI5 versions grouped by maintenance status as listed here.
  - If the source system during generation is an ABAP on Premise system, then the **default** version selected in the dropdown will be equal to the version of SAPUI5 version on that ABAP system where possible.

  Note

  For an application generated with the OData V4 data source, the list of SAPUI5 versions supported is limited to the most recent ones.

- Add Deployment Configuration to an Existing MTA Deployment File.

  If a project is generated inside an app router configuration project that has an MTA file, then it will use the deployment configuration by default. In this case, you can still select **No** and skip the deployment configuration step.

- Add deployment configuration. The default value is **No**.

  Provide values to the prompts and click **Finish**.

- Add FLP configuration. The default value is **No**.

  Provide values to the prompts and click **Finish**.

- Configure advanced options. The default value is **No**.

5. Click **Finish** to finalize generating the application.

Supported Floorplans

This is a list of supported floorplans that gives you the ability to create SAP Fiori elements applications based on it.

- **List Report Object Page**.

  With the **List Report Object Page**, users can view and work with a large set of items. This floorplan offers powerful features for finding and acting on relevant items. It's often used as an entry point for navigating to the item details, which are shown on the object page.

  For more information, see List Report or Object Page.

- **Worklist**.

  The worklist displays a collection of items that the user needs to process. Working through the list usually involves reviewing details of the items and taking action. In most cases, the user has to either complete a work item or delegate it.

The focus of the worklist floorplan is on processing the items. This differs from the list report floorplan, which focuses on finding and acting on relevant items from a large dataset.

You can use any **List Report and Object Page** options to configure **Worklist**

For more information about **Fiori Design Guidelines**, see Worklist Floorplan

> **Note**
>
> For **Worklist** floorplans using an `OData V4` data source, only `SAPUI5` versions 1.99 and above are supported.

For information on what is supported in **Worklist** for `OData V2`, see Floorplan properties.

- **Analytical List Page**.

  The **Analytical List Page** offers a unique way to analyze data step by step from different perspectives to investigate the root cause of any deviations, spikes, and abnormalities through drilldown, and to act on transactional content. All this can be done seamlessly within one page. The purpose of the analytical list page is to identify problem areas within datasets or significant single instances using data visualization and business intelligence.

  Visualization helps users to recognize facts and situations, and to reduce the number of interaction steps needed to gain insights or to identify significant single instances. Chart visualization increases the productivity of use and enables users to spot relevant data more quickly.

  The main target group are users who work on transactional content. They benefit from fully transparent business object data and direct access to business actions. In addition, they have access to analytical views and functions without having to switch between systems. These include KPIs, a visual filter where filter values are enriched by measures and visualizations, and a combined table or chart view with drill-in capabilities (hybrid view). Users can interact with the chart to look deep into the data. The visualization enables them to identify spikes, deviations, and abnormalities more quickly, and to take appropriate action right away.

  For more information, see Analytical List Page

  > **Note**
  >
  > For **Analytical list page** floorplans using `Odata V4` data source, only `SAPUI5` versions 1.90 and above are supported.

- **Overview page**.

  The overview page is a data-driven SAP Fiori application type and floorplan that provides all the information the user needs on a single page, based on the user specific domain or role. It allows the user to focus on the most important tasks, and view, filter, and react to the information quickly.

  Each task or topic is represented by a card or a content container. The overview page acts as a UI framework for organizing multiple cards on a single page.

  The overview page is based on SAP Fiori elements technology, and uses annotated views of app data, meaning that the app content can be tailored to the domain or SAP Fiori elements role. Different types of cards allow you to visualize information in an attractive and efficient way.

  For more information, see Overview Page.

- **Form entry object page**.

  The form entry object page allows users to create an application with an object page and a generated form. The object page floorplan enables end-users to provide data entry in the generated application.

  For more information, see Object Page.

- **Flexible programming model**.

The flexible programming model makes it easy for you to extend apps based on SAP Fiori elements for OData V4. You can use any SAPUI5 coding or controls in extension points, and take advantage of the provided building blocks.

For more information, see Flexible Programming Model.

Data Source

This section provides information on how to connect your application with a data source during generation.

ℹ Note

When running SAP Fiori tools in VS Code, you can save the connection information to a remote system and find it under the **SAP Systems** view. For more information, see Managing System Connection.

Connect to an SAP System

- **Connect to an SAP system using VS Code.**

  In either case, you can create a new system to connect to, or select one of the saved systems you may have already used.

  Adding a new system

  1. **Enter a system name** that you use to save the connection details for either an on-premise **SAP ABAP system** or SAP Business Technology Platform system.
  2. **Select a service key**.

     For the **SAP ABAP system** hosted in the SAP Business Technology Platform, you must provide a service key that contains the key information for the required **SAP ABAP system**. This service key should be provided by your administrator for the selected SAP ABAP system. Once this information is available, a browser tab launches and prompts you to authenticate against the system.

     For more information on how to create a service key, please see Create Service Keys Using the Cockpit.

     #### ℹ Note

     Once you have authenticated your user for the new system in the browser, SAP Fiori application generator shows you the list of OData services available for the user you have used to log in. Please see the **Service** dropdown with the title **Service (for user [<USERNAME>])**.

     For an on-premise **SAP ABAP system**, you need to provide the system URL and optional client ID, along with the authentication details for that system if required.

     #### ✤ Example

     https://someurl:12345, client: 010

  In both scenarios, you can store the system details in the secure storage of your operating system:

  - Microsoft Windows: Credential Manager
  - macOS: Keychain

  Saving the system in this way ensures that you do not need to continually provide these details for generating an application or running it locally.

  The saved connections can be viewed and deleted in the **SAP Systems** view in VS Code.

If you want to update the connection, we recommend that you delete the system from the **SAP Systems** view and recreate it in the project generator. For more information, see Managing System Connection.

ⓘ Note

When connecting for the first time, only the **New System** option is available.

Using a saved system

1. Select a previously saved system.

   The wizard skips the authentication if your saved system credentials are still valid.

2. If saved authentication details are invalid, you will be prompt to reauthenticate. For example, if your password is expired.

- **Connect to an SAP system using SAP Business Application Studio.**

  When using the SAP Fiori application generator in SAP Business Application Studio, you can select from a list of destinations that are configured for the SAP Business Application Studio instance. The generator automatically retrieves the available destinations to select from. If you do not have the correct access to use the destination endpoint, an error occurs.

  The destinations are expected to have a catalog service that provides a list of V2 and V4 OData services that are available.

  If you want to use a destination to reference a service URL endpoint, see Connect to an OData Service with a customized URL.

ⓘ Note

In SAP Business Application Studio, to create SAP Fiori elements application in SAP BTP ABAP environment, you must be assigned to one of the following roles:

- OrgManager
- SpaceManager
- SpaceDeveloper

Connect to an OData Service with a Customized URL

If the OData endpoint that you want to use in your application can't be accessed directly, you can set it up as a destination and directly reference it in the generator. To do so, perform the following steps:

1. In SAP Business Application Studio, launch the SAP Fiori application generator and select the required template.
2. Select **Connect to an OData Service** from the data source drop-down list.
3. For the data source URL field, use the destination name followed by `.dest`. In this case, SAP Business Application Studio should be able to route to your service with the destination name.

Example

If the URL defined in the Destination is `https://someurl.com/someservice` with the destination name "MyDestination", the following URL will be used in the SAP Fiori application generator:

```
https://MyDestination.dest/someservice
```

Connect to an OData Service

Enter the OData endpoint URL to generate your application.

- All OData endpoints that are either **authenticated** or **unauthenticated** are supported.
- The OData endpoint must be the correct version of the template that you've selected. For example, a V2 endpoint must be provided for a V2 template. The wizard informs you in case of any mismatch between the OData version and the template version

ℹ Note

If necessary, the system prompts you to provide your name and password.

Upload a Metadata Document

Upload a `metadata XML` file that you want to use. Now, you can generate the application without relying on a backend service being available.

- Only EDMX format is supported for the `metadata XML` file.
- Once the `metadata XML` file is validated, you can select the required entity options for the application.

ℹ Note

Using the `metadata.xml` file restricts the generated application to only mock data.

Connect to SAP API Business Hub

When users don't have their data source available, they can generate an application with the SAP API Business Hub. This data source is only intended to support the development and should be replaced with a real one before going live.

ℹ Note

Ensure that you logged in to https://api.sap.com/ at least once before connecting to SAP API Business Hub.

- In the **Data Source and Service Selection** wizard page, select "Connect to SAP API Business Hub" from the **Data source** drop-down list.
- When the SAP API Business Hub option is selected, a list of predefined services relevant to different industries appears. Select a service that you want to generate an application with. For example, Just-In-Time Calls, Transaction Classifications, Content, Request of Quotation, and more.
- Once the service is selected, two more fields appear for authentication purposes: **Enter your Username** and **Enter your Password**.
- Fill in the fields and click **Next** to proceed with the application generation.

ℹ Note

You cannot deploy applications that use the SAP API Business Hub, as this data source is intended for local development only.

For more information, see https://api.sap.com/.

Use a Local CAP Project

You can either select a local SAP Cloud Application Programming Model (CAP) project that has been detected in your workspace, or manually select the CAP project folder path and the generator will retrieve the services that are defined for that project.

ℹ Note

A local CAP Project data source is limited to the List Report Object Page and Analytical List Page templates that are based on the OData V4. This option is not available for the Worklist and Overview Page templates.

ℹ Note

OData V2 Support for CAP - While CAP defaults to OData V4, the latest protocol version, some projects need to fallback to OData V2, for example, to keep using existing V2-based UIs. SAP Fiori tools does not recommend and support having both OData V2 and OData V4 applications in the app folder within the CAP project. In case, you have a requirement to create both OData V2 and ODataV4 applications, it is recommended that you generate the OData V2 app outside of the CAP project.

For more information about CAP services, see: https://cap.cloud.sap/docs/about/.

SAP Fiori tools support two types of SAP CAP Projects:

SAP CAP Node.js Project

Project Prerequisites

- Node.js
- Sample Projects:
  - CAP Node.js Teched2020 .
  - Cloud CAP Samples .

    ### ℹ Note

    If you use one of the sample projects, ensure the workspace root is set to the cloned repository and execute `npm install`.

SAP CAP Node.js Project steps

1. Perform the steps identified in the SAP Fiori Elements section.
2. On the **Data Source and Service Selection** wizard page, in the **Data source** drop-down list, select **Use a Local CAP Project**.
3. If there are **CAP projects** detected in your workspace, you can then choose from the list of **CAP projects** that have been found. If your **CAP project** is not in the list, select **Manually select CAP project folder path** and browse to your **CAP project** location. If the generator is unable to find any local **CAP projects** in your workspace, you can provide the **CAP project** folder path manually.
4. In the **OData service** drop-down list, select a service, and click **Next**.
5. On the **Entity Selection** wizard page, select the main entity from a drop-down list.

   ### ℹ Note

   The properties of this entity are used to display data on the List Report from the entity collection.

6. Leave the value for the **Navigation Entity** field as **None** and click **Next**.

7. On the **Project Attributes** wizard page, add the required attributes to the application project. For example:
    - **Module name**: `incidents`.
    - **Application title**: `My Incidents`.
    - **Application namespace**: `sap.fe.demo`

    For the rest of the attributes, keep the default values.

    For more information on the Project Attributes wizard page, see SAP Fiori Elements Application.
8. Click **Finish** to start the app generation.

SAP CAP Java Project

Project Prerequisites

- Ensure `cds`, `java`, and `mvn` are installed on your computer to be able to execute these commands in the terminal and identify their version numbers:

  ```
  cds --version
  java --version
  mvn --version
  ```

  For more information, see Setting Up Local Development in the CAP Java documentation.
- Sample Projects:
    - CAP Samples for Java .

      > **Note**
      >
      > If you use one of the sample projects, confirm that all system requirements are met. Execute `mvn` command:
      >
      > ```
      > cd cloud-cap-samples-java
      > mvn spring-boot:run
      > ```

  It takes some time for the command to install dependencies, build, and start the project. After the command is executed, click the `http://localhost:8080` link to open a browser and see the services, such as `admin` or `browse`. You can cancel the command by entering `Ctrl + C` in the terminal.

SAP CAP Java Project steps

1. Perform the steps identified in the SAP Fiori Elements section.
2. On the **Data Source and Service Selection** wizard page, in the **Data source** drop-down list, select **Use a Local CAP Project**.
3. If there are **CAP projects** detected in your workspace, you can then choose from the list of **CAP projects** that have been found. If your **CAP project** is not in the list, select **Manually select CAP project folder path** and browse to your **CAP project** location. If the generator is unable to find any local **CAP projects** in your workspace, you can provide the **CAP project** folder path manually.
4. In the **OData service** drop-down list, select a service, for example, CatalogService, and click **Next**.
5. On the **Entity Selection** wizard page, select the main entity from a drop-down list.

> **Note**

The properties of this entity are used to display data on the List Report from the entity collection.

6. Leave the value for the **Navigation Entity** field as **None** and click **Next**.
7. On the **Project Attributes** wizard page, add the required attributes to the application project. For example:
    - **Module name**: `books`.
    - **Application title**: `Books`.

    For the rest of the attributes, keep the default values.

    For more information on the **Project Attributes** wizard page, see SAP Fiori Elements Application.
8. Click **Finish**.
9. In the terminal, execute the command `mvn spring-boot:run`.
10. Open the file `app/books/README.md` and find the link leading to the application, such as `http://localhost:8080/books/webapp/index.html`.
11. Click the link to the application.
12. When prompted, enter the username and password. For example, `user/user` or `admin/admin`.

    Note

    You can check the terminal output to see the username and password credentials to be used.

The SAP Fiori Launchpad sandbox with a newly created application appears.

Floorplan properties

Once the data source is supplied, you can customize the application by selecting from the list of entities in the data source.

Generic options Across Multiple Floorplan Types

- **Main Entity**. Represents the entity set that is used to populate the main content area of the list page. For a parametrized entity set, use a result entity set name instead.
- **Navigation Entity**. Represents association from the main entity to navigate to apps

Worklist Page V2 specific properties

- The worklist does not contain a smart filter bar. The search field is available in the table toolbar.
- **Variant management**. By default, variant management is hidden. You can customize the worklist to provide variant management at table level. To do so, set the **variantManagementHidden** flag to false in the `manifest.json`. You can enable page level variant management by setting **smartVariantManagement** to true and the **variantManagementHidden** flag to false in the `manifest.json`. Variants can also be shared. The **Execute on Select** action is not available.
- **Smart table**. The multiselect function is enabled for all tables. If there are only line item actions, a no-selection table is enabled. The **Export to Microsoft Excel** feature is not available. The default table type is **responsive**. The table title contains the row count. A fixed layout and growing using the scrolling function is enabled.

Analytical List Page V2 specific properties

- **Table type**. Defines different table types supported in the **Analytical List Page**.

- **Allow multi select**. Enables a checkbox for selecting multiple items in a table to be displayed. This setting is effective only in the case of defined actions either through an annotation or manifest.
- **Auto hide**. Determines chart or table interaction. If it is set to true, the chart acts as a filter for a table. If it is set to false, the matching table rows are highlighted but the table is not filtered.
- **Enable smart variant management**. Enables page-level variant.

Analytical List Page V4 specific properties

- **Selection mode**. Defines different row selection modes for the generated SmartTable.

Overview Page

For each floorplan type, except for the **Overview Page**, you can select a main entity from the drop-down list to be used in the generated application.

Select the entity to be used as the filter type from the drop-down list of entities, mandatory field. After the main entity is selected, a list of navigation entities that are filtered depending on the main entity is displayed.

ℹ Note

The navigation entity is an optional field.

SAPUI5 Freestyle

Generating an application SAPUI5 freestyle.

ℹ Note

In SAP Business Application Studio, use SAP Fiori elements Dev Space.

1. Select Supported Templates, click **Next**.
2. Select Data Source, click **Next**.
    - **Connect to an SAP system**.
    - **Connect to an OData service**.
    - **Connect to SAP API Business Hub**.
    - **Use a local CAP Node.js project** (Available for the SAP Fiori Worklist Application OData V4 template only).
    - **Upload a Metadata Document**.
    - **None** (Available for **Basic** application only).

ℹ Note

If username and password are required, enter these credentials and click **Login**.

3. Select Template Properties.

4. Add project attributes:

> **Note**
>
> Mandatory fields are prefilled with default text.

- **Module Name** (Required). Must be alphanumeric and **can't contain spaces**. The generated NodeJS application uses the module name as its package name. It's used as the folder name of the generated application.
- **App Title** (Required). The title in the header of the generated application.
- **App Namespace** (Required). The SAPUI5 project namespace to be used. Must start with a letter and contain letters, digits, and periods only.
- **Description**. The text description of the application.
- **Project Folder** (Required). The parent folder in which the new application is generated. The new application is generated in a new folder with the module name. If a folder with the same name already exists, the user must choose a new module name.
- **Minimum SAPUI5 version**. From the drop-down list, select the minimum SAPUI5 version that the application will require.
  - The dropdown will show the list of available versions of SAPUI5, with the current **default** version being preselected. The dropdown will list SAPUI5 versions grouped by maintenance status as listed here.
  - If the source system during generation is an ABAP on-Premise system, then the **default** version selected in the dropdown will be equal to the version of SAPUI5 version on that ABAP system where possible.

  > **Note**
  >
  > For an application generated with the OData V4 data source, the list of SAPUI5 versions supported is limited to the most recent ones.

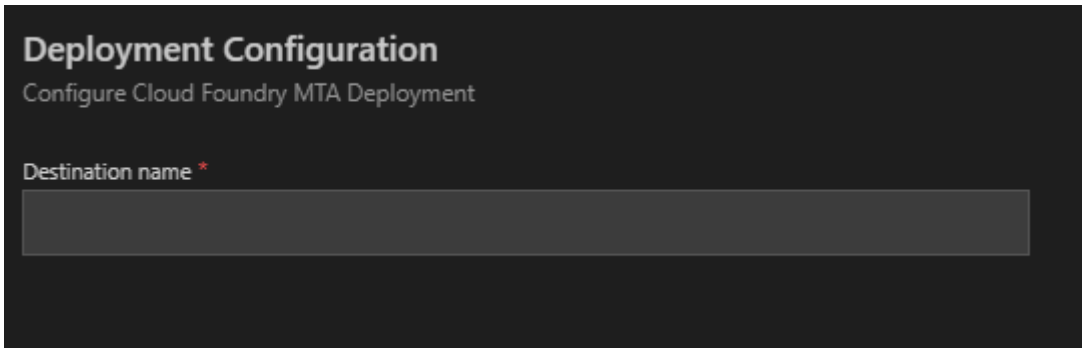
- **Advanced Options**:
  - Select the SAPUI5 theme:
    - SAP Quartz Light
    - SAP Belize
    - SAP Quartz Dark

    > **Note**
    >
    > SAP Quartz Dark is only available in SAPUI5 versions 1.72 and later.

- **Adding to MTA project**.

  Generated project is added to the existing MTA config at the time of application generation. The project creates in a subfolder with MTA config.

  Enter a mandatory destination name.



5. Click **Next** to generate the application.

Supported Templates

This is a list of supported templates that enable you to create SAPUI5 freestyle applications.

- **Basic**

  The Basic Template is intended for those developers who want to create their SAPUI5 app from scratch. With this basic template, a blank canvas is available to start coding right away. The basic file structure is set up according to our best practices.

- **SAP Fiori Worklist Application**

  The **Worklist Template** template implements a typical worklist template, one of the patterns that is specified by the SAP Fiori Design Guidelines.

  A worklist displays a collection of items to be processed by the user and usually involves reviewing details of a list item and taking action. If the data needs to be organized into columns or the overview of the items is more important than displaying the item details directly, this template can be used as a starting point.

- **SAP Fiori List-Detail Application**

  The **List-Detail Template** template implements a flexible column layout, one of the design patterns that is specified by the SAP Fiori Design Guidelines.

  The flexible column layout is a layout control that displays multiple templates on a single page. This ensures faster and more fluid navigation between multiple templates than the usual page-by-page navigation. The flexible column layout offers different layouts with up to three columns. In the template, two columns can be used, which are list and detail.

  > ⚠ **Caution**
  > The SAP Fiori Worklist and SAP Fiori List-Detail freestyle templates are deprecated. It's recommended to use the Custom page SAP Fiori template based on the flexible programming model as an alternative.

  > ⚠ **Caution**
  > Currently, SAP Fiori tools support the development of SAP Fiori elements and SAPUI5 freestyle applications with minimum SAPUI5 versions 1.65 or higher.

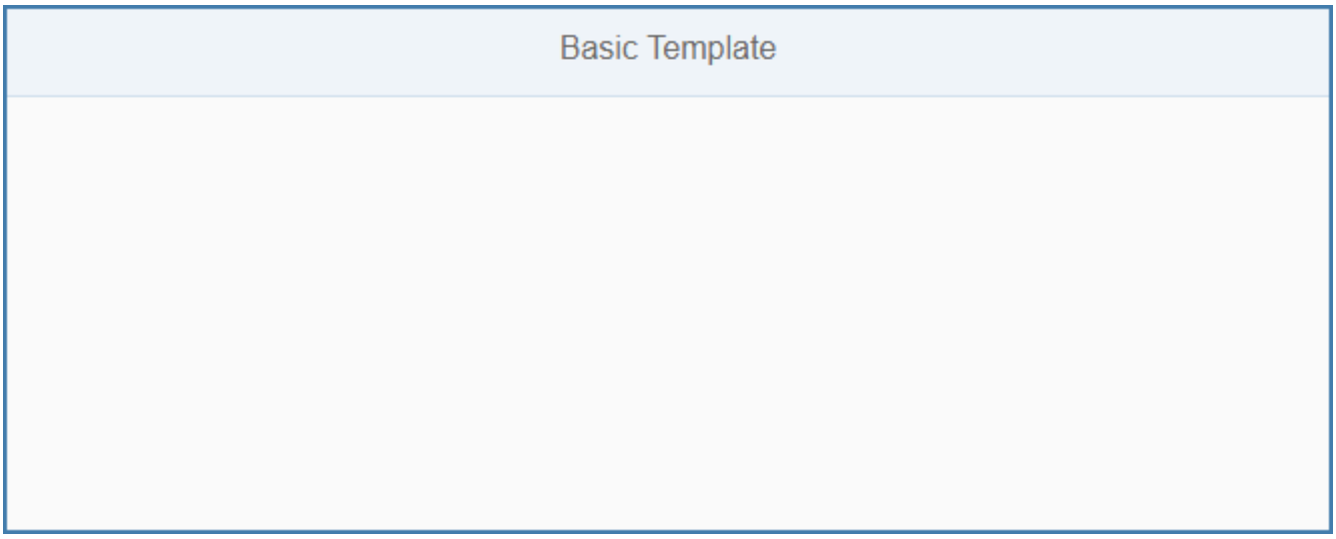
Basic Template

The basic template is intended for all developers who want to start developing their own SAPUI5 app from scratch.

With this basic template you have a blank canvas to start coding right away. The basic file structure is set up according to our best practices.

ℹ Note

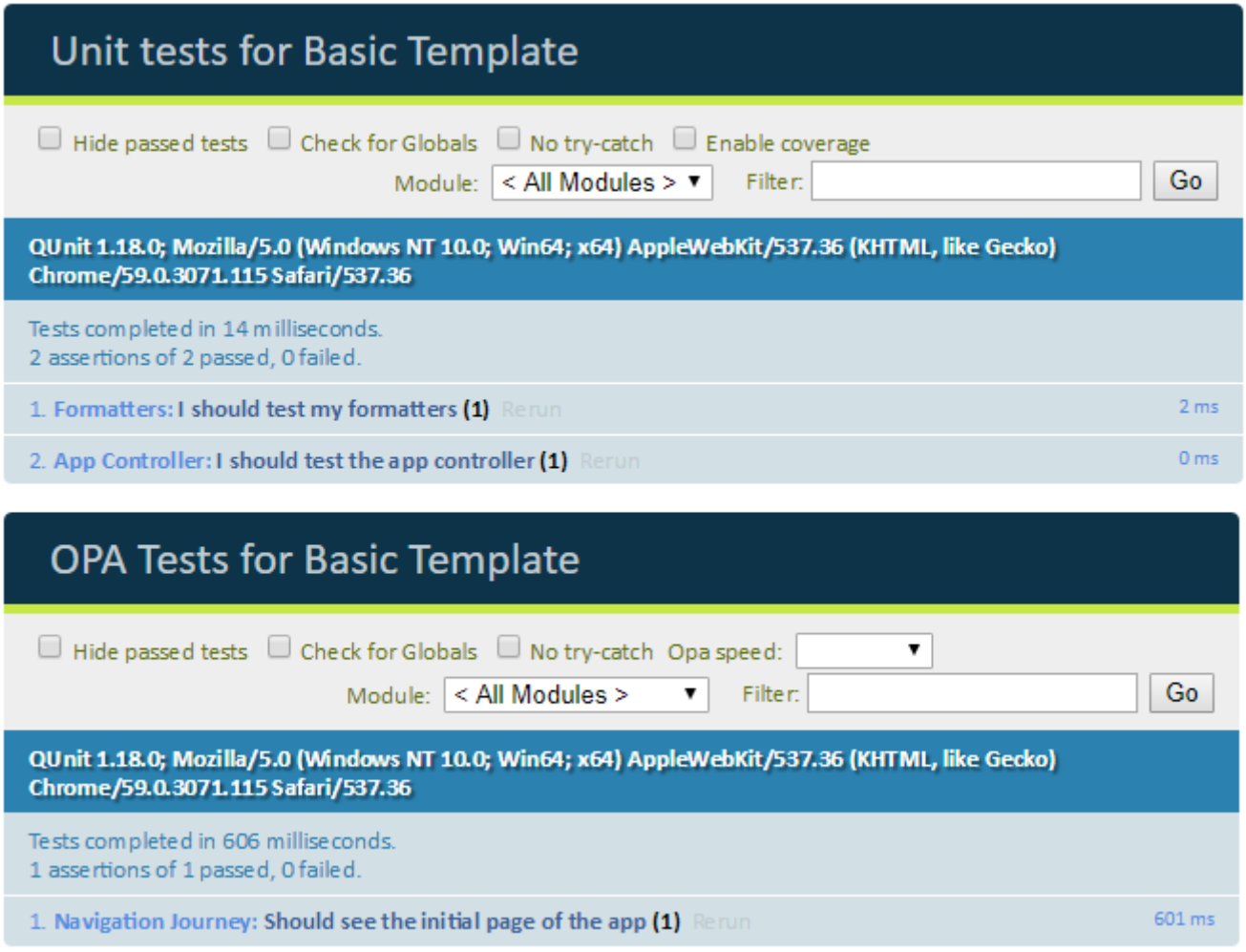
This template does not include SAP Fiori launchpad features and is intended for standalone use. If you want to convert it to a launchpad app you have to add some features manually, such as the **Save as Tile** feature.



Screenshot of the Basic App

The `index.html` file defines the page that is displayed when the app is started. It is located in the `webapp` folder. It contains an XML view with a header and a title from the `sap.m` library as a starting point. You can easily modify the app to add more functionality.

Integrated Tests



An important best practice is to have unit and integration tests for your app. With this template, we have included sample tests that you can use: Tests on formatters and the app controller are the basic tests any app should cover. You can find them in the `test` subfolder of the `webapp` folder.

Where Can I Find the Basic Template?

You can find the template in the following places:

- **Basic** template in SAP Fiori tools.
- `openui5-basic-template-app` in the SAP Repository on GitHub .

  For more information about how to clone or download the template from GitHub, refer to the template documentation on GitHub .

List-Detail Template

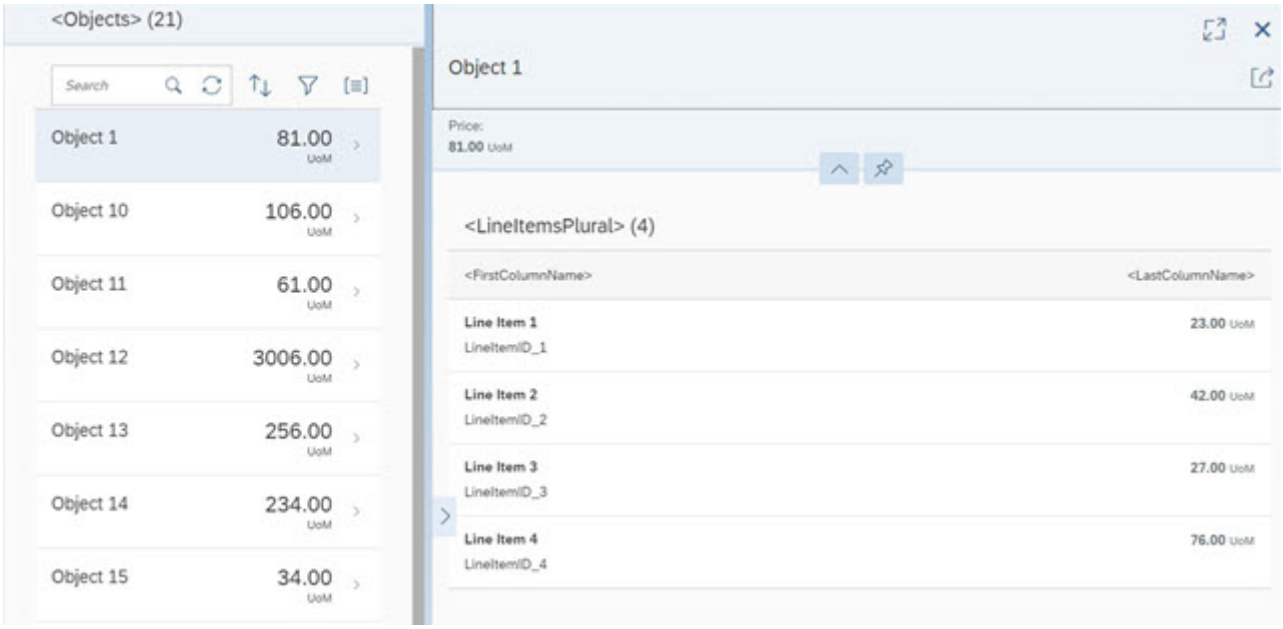
The **SAP Fiori List-Detail Application** template implements a flexible column layout, one of the design patterns that is specified by the SAP Fiori design guidelines .

The flexible column layout is a layout control that displays multiple templates on a single page. This allows faster and more fluid navigation between multipletemplates than the usual page-by-page navigation. The flexible column layout offers different layouts with up to three columns. In the template, we use two colums (list and detail). For more information about flexible columns and list-detail apps, see the SAP Fiori Design Guidelines.

ℹ Note

You have two options: You can use this template to build an **app for the SAP Fiori launchpad** or to build **standalone apps**.

- If the app runs in SAP Fiori launchpad it also contains additional features like **Save as Tile** or **Share in SAP Jam** that depend on SAP Fiori launchpad at runtime. This app cannot be run standalone, meaning no `index.html` file is created but only files for testing the app in the SAP Fiori launchpad sandbox.
- Only standalone apps contain an `index.html` file that is used to start the app.



Screenshot of the List-Detail App

The main control of this app is the `sap.f.flexibleColumnLayout` control. This control first displays only the **List** view with a list of objects. When the user selects an object in the list, the **Detail** view is displayed on the right side, showing the details for the selected item.

The **List** view shows the current number of items and a search field that can be used to search through the list items. The number of items are updated automatically and the search filters for a preconfigured field of the list. Functionality for sorting, filtering, and grouping the list is also included in the template as well.

The **Detail** page contains a dynamic page header displaying more details for the selected object, an `sap.m.Overflowtoolbar` that can be enriched with custom content, and a table of line items that are associated to the selected object in the data model.

The list and the line item table are set to `growing` mode so that initially only the first few items are displayed for performance reasons. Using the `scrollToLoad` feature, the user can display more items by scrolling down or pressing the trigger at the end of the list.

We use the semantic `MasterPage` and `DetailPage` controls for the content aggregations of the `sap.f.FlexibleColumnLayout` control. A `SemanticPage` is an enhanced `sap.f.DynamicPage` that contains controls with semantic-specific meaning and displays them according to the SAP Fiori design guidelines. For more details about semantic controls, see the sample, `sap.f.semantic.SemanticPage`, in the Demo Kit.

Where Can I Find the List-Detail Template?

You can find the template in the following places:

- **SAP Fiori List-Detail Application** template in SAP Fiori tools
- `openui5-masterdetail-app` in the SAP Repository on GitHub

  For more information on how to clone or download the template from GitHub, refer to the template documentation on GitHub .

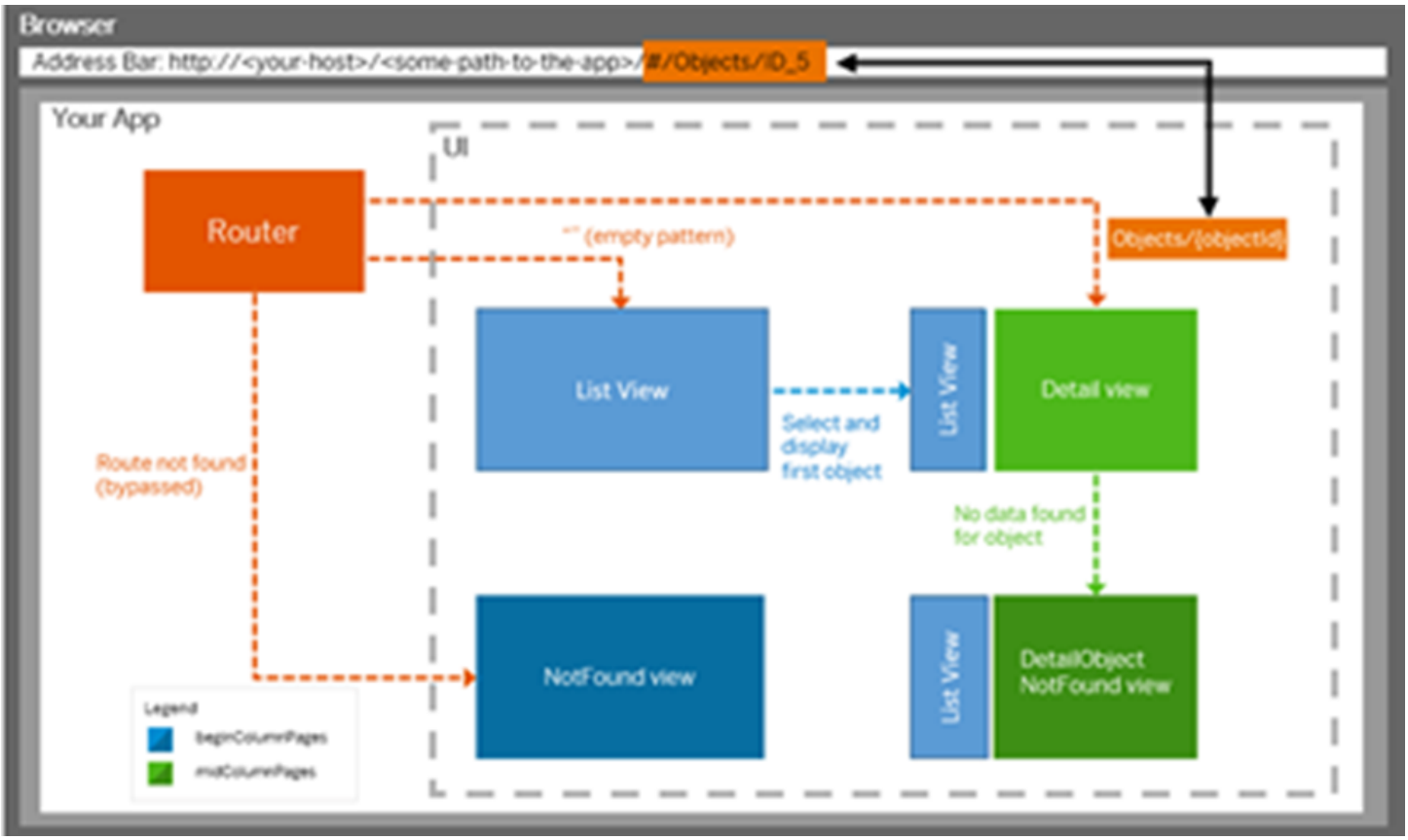
How Do I Enhance the Template?

You can find more information about the possibilities of object pages at SAP Fiori Design Guidelines - Object Page.

Navigation

The navigation flow of the List-Detail app considers both the **List** and **Detail** pages, and is therefore slightly more complex than a typical full-screen scenario.

With an empty hash in the URL, only the master view is shown initially. When the user enters the app with an object id in the hash, both views are loaded at the same time, and methods in the controller logic make sure that the pages are in sync. Additional **not found** pages display a message to the user in case of any navigation errors that occur for the master and the detail page.



Navigation Flow of the List-Detail App

The two main views **List** and **Object** each have a route and two targets configured. When the route matches the URL, both targets are displayed and the corresponding views are created. The target master puts the created view in the `beginColumnPages` aggregation of the `sap.f.FlexibleColumnLayout` control. All other targets put their created views in the `midColumnPages` aggregation. For more information, see Routing and Navigation.

Here is a sample implementation for navigating from the **List** to the **Object** page. The below `_showDetail` method is called by the `selectionChange` event handler of the `sap.m.List` control. We need to change the layout parameter of the `sap.f.FlexibleColumnLayout` to `TwoColumnsMidExpanded` and navigate to the `object` route. Then, we extract the current ID of the object pressed by using its binding context. We supply this parameter to the mandatory `objectId` parameter and pass it to the `navTo` function, as described in the `sap.ui.core.routing.Routing#navTo` section in the **API Reference** in the Demo Kit and shown here:

```
...
/**
  * Shows the selected item on the detail page
  * On phones an additional history entry is created
  * @param {sap.m.ObjectListItem} oItem selected Item
  * @private
  */
_showDetail : function (oItem) {
      var bReplace = !Device.system.phone;
      // set the layout property of FCL control to show two columns
      this.getModel("appView").setProperty("/layout", "TwoColumnsMidExpanded");
      this.getRouter().navTo("object", {
         objectId : oItem.getBindingContext().getProperty("ObjectID")
      }, bReplace);
},
...
```

After calling `navTo`, the hash of the browser is updated, and you get an event on the `DetailController` when the route `object` matches the current hash. In the `_onObjectMatched` handler that we register in the `init` method of the controller, we extract the `objectID` from the event arguments and create a valid model path with the help of the `createKey` method of our OData model. We then bind the data to the view:

```
...
/**
  * Binds the view to the object path and expands the aggregated line items.
  * @function
  * @param {sap.ui.base.Event} oEvent pattern match event in route 'object'
  * @private
  */
_onObjectMatched : function (oEvent) {

      var sObjectId =  oEvent.getParameter("arguments").objectId;
      this.getModel("appView").setProperty("/layout", "TwoColumnsMidExpanded");
      this.getModel().metadataLoaded().then( function() {

        var sObjectPath = this.getModel().createKey("Objects", {
          ObjectID :  sObjectId
        });
        this._bindView("/" + sObjectPath);
      }.bind(this));
},
...
```

notFound (similar to an HTTP 404 "not found" status code)

The **not found** pages are implemented using an sap.m.MessagePage. They display an error message according to the SAP Fiori UX specifications. There are different "not found" cases that each have a separate target and a **notFound** view.

If you have the following URL, no route will match: `index.html/#/thisIsInvalid`. This means that the **notFound** view will be displayed, as the target **notFound** is defined in the bypassed section.

The code sample below shows the relevant parts of the configuration. In addition, we set the layout property for the `sap.f.FlexibleColumnLayout` to `OneColumn` in the controller of the **notFound** page so that only a single column is displayed in this case. For a full implementation of a **not found** page, see Catch Invalid Hashes.

```
"routing": {
    "config": {
    …
    "bypassed": {
    "target": "notFound"
    }
    }
    …
    "targets": {
    …
    "notFound": {
      "viewName": "NotFound",
      "viewId": "notFound"
    }

}
```

detailObjectNotFound

If the object route matches – an ID is passed (for example `#/Objects/1337`) but the back end does not contain an object with the ID `1337`, then you need to display the **detailObjectNotFound** page. This is achieved by listening to the "change" event of a binding. Inside this, you check if there is no data and tell the router to display the **detailObjectNotFound** target, as shown in the sample code below:

```
// inside of a controller
this.getView().bindElement({
    path: "/Objects/1337",
    change: function () {
      // there is no data
      if (!this.getView().getElementBinding().getBoundContext()) {
          this.getRouter().getTargets().display("detailObjectNotFound
");
      return;
                                }
      // code handling the case if there is data in the backend
      …
};
});
```

Busy Indication

The List-Detail application implements a busy indication concept as specified by the SAP Fiori Design Guidelines. Calling the application will result in the following:

- Only initially a global busy indicator is displayed that overlays the whole application until the metadata of the service is loaded.
- Afterwards, a local busy indicator is displayed on the list and the detail page.
- When the detail page is loaded, the line item table on the detail page is set to busy until the line items are loaded with a separate service call.
- When controls are loading additional data or getting refreshed, a local busy indication is displayed automatically.

By default, the busy indicator delay is set to one second for all controls. This would first show the UI for a second, then show a busy indication until the data is loaded. To avoid this behavior initially and show the busy indicator immediately without delay, the following concept is implemented in the application: The `busyIndicatorDelay` and `busy` properties of certain controls (`AppView`, `List` on the **List** page, `DetailPage` and `Table` on the **Detail** page) are bound to the local view model and manipulated in the controllers of the application. The delay is initially set to **0** for displaying the busy indicator immediately, and reset to the previous value after the initial loading is done.

You can simulate server delays to test this implementation running with mocked application data by using the URL parameter `serverDelay=true` in the hash. The default is set to **1000ms**.

ℹ Note

You can find more information about busy indicators, busy states, and busy handling in general in the SAP Fiori Design Guidelines.

Model Instantiation

The application configures several data models that are used throughout to update the views or to store additional configuration options.

The service model and the resource bundle are instantiated automatically by the applications component during startup and described in the first section. The local view models and helper models such as the device model are set up as JSON models and described in the second section.

Automatic Model Instantiation

The templates instantiate the service and resource model automatically using the following configuration entries in the descriptor. When the component of the app is initialized, these models will be made available under the configured name throughout the app.

An external service is defined in the `dataSources` section of the `sap.app` namespace. In the example shown below, we configure an OData V2 model and the alias `"mainService"` in the `manifest.json` descriptor file:

```
{
        …
        "sap.app": {
                …
                "i18n": "i18n/i18n.properties",
                …
                "dataSources": {
                        "mainService": {
                                "uri": "/here/goes/your/serviceUrl/"
                                "type": "OData",
                                "settings": {
                                        "odataVersion": "2.0",
                                        "localUri": "localService/metadata.xml"
                                }
                        }
                },
                …
        },
        …
}
```

ℹ Note

If you use the OData V4 template, you set the `odataVersion` accordingly.

In the models section of the `sap.ui5` namespace we define two models that will be instantiated automatically. The resource model is a named model (i18n) and the OData model is the default model so it has no name. The OData model also receives additional URL parameters via the `metadataUrlParams`. The parameters `sap-server`, `sap-client`, and `sap-language` are passed to the service automatically by SAPUI5, as shown in the following `manifest.json` code snippet:

```
{
        …
        "sap.ui5": {
                …
                "models": {
                        "i18n": {
                                "type": "sap.ui.model.resource.ResourceModel",
                                "settings": {
                                        "bundleName": "sap.ui.demo.masterdetail.i18n.i18n",
                                        "supportedLocales": [""],
                                        "fallbackLocale": ""
                                }
                        },
                        "": {
                                "dataSource": "mainService",
                                "preload": true
                        }
```

```
                },
                …
        }
}
```

ℹ Note

Before SAPUI5 version 1.30, all models were defined and instantiated in the component's `init` method. We recommend removing all manual model creation code and switching to the automatic model instantiation instead. The "device model" however is still a local model that has to be instantiated manually.

Additional Models for the App

The following models are created as local JSON models in the app and can be referenced by its model name where needed:

- **device**

  The device model provides an easy access to the sap.ui.Device API and is used to configure certain view settings according to the user's device.

- **Fiori launchpad (FLP)**

  The FLP model is a helper module to configure SAP Fiori launchpad integration and is used to control the sharing options of the app.

- **masterView**

  A local view model for the `master` view that stored configuration options that are bound to controls in the view.

- **detailView**

  A local view model for the `detail` view that stored configuration options that are bound to controls in the view.

- **appView**

  A local view model for the `app` view that stored configuration options that are bound to controls in the view.

Related Information

Resource Bundle API
class sap.ui.model.odata.v2.ODataModel
OData V2 Model
Descriptor for Applications, Components, and Libraries

List Filtering

You can use the following best practices when implementing search, sorting, filtering and grouping functions for a main list in your List-Detail apps.

A search field is displayed in the main list to filter the list items for a custom keyword. In the header toolbar of the master list, options for sorting, filtering, and grouping are displayed. When searching or using one of the options in the header, the list content is updated automatically, and the search result is displayed.

All four options adjust the main list content (search, sort, filter, group) and are managed and applied in the logic of the master controller. This section describes the implementation details for these four options.

Search

The search is implemented in a manual mode and the list operation mode is "server". This means that the search has to be triggered explicitly by pressing enter or the search button, and the results are always fetched from the server.

The search function is implemented using the standard SAPUI5 `sap.ui.model.Filter` objects. The options are added to an internal state object of the controller and applied together with the filters that can be selected in the filter options. The type of these filters is "Application", and these filters are added on top of the predefined filters from the framework of type "Control".

The **Search** field also displays a **Refresh** button. Pressing this button triggers a simple refresh for the list binding.

Sorting, Filtering and Grouping

Sorting, filtering, and grouping can be implemented by using a semantic button that opens a `sap.m.ViewSettingsDialog` containing options for sorting, grouping, and filtering.

The event handlers that are called when selecting a sorting and grouping option are similar. They are implemented as an XML fragment with a `sap.m.ViewSettingsDialog` in a fragment. Therefore, we process the selected options in the handler of the dialog's `confirm` event. The event handlers create a `sap.ui.model.Sorter` object on the **key** field of the selected item. For the grouping functionality, a custom grouper is loaded and applied to the selected entry. Both sorting and grouping options are applied together on the binding of the main list. A `sap.ui.model.Filter` object is created for each filter option that has been selected in the dialog and applied together with the search option on the main list.

The filter message is automatically updated with the chosen filter texts. It is displayed on top of the main list and can be clicked to reopen the filter settings.

Related Information

SAP Fiori Design Guidelines: List

Send Email

The **Send Email** feature is a sharing option that can be found in the share menu of each view.

This feature simply triggers a `sap.m.URLHelper` action that will show a new email with preconfigured texts in the default client of the user.

The placeholder texts are located in the resource model and can be adjusted to your use case. The texts already include the current context and the current location. The texts may vary for each view, therefore they are configured with the local view model and updated when the business object context has changed.

For more information about `sap.m.URLHelper`, see the sample in the Demo Kit.

Testing

The templates include basic testing features, unit tests as well as integration tests for a basic test coverage of the initial app. The tests are written independently of the actual data displayed in the app.

The `webapp` folder of the template app contains a `test.html` file which serves as an overview for the different test pages. You can run the app with or without mock data and run the unit and integration tests. This section describes which application tests are provided and how they are structured.

Integration Tests

The integration tests shipped with the template cover all basic functionality and provide several "journeys". Journeys include a series of OPA tests that belong to the same functionality and should be executed together. Some of the journeys are implemented for both phone and desktop use cases to test device-specific interaction steps:

- `BusyJourney` / `BusyJourneyPhone`: This journey tests the busy indication features of the app for phone and other devices.
- `NavigationJourney` / `NavigationJourneyPhone`: This journey will trigger user interactions and navigate through the application. The routing configuration, basic navigation events, and error handling are tested here.
- `NotFoundJourney`/`NotFoundJourneyPhone`: Several "not found" cases of the application are tested here. Faulty navigation scenarios are introduced intentionally to simulate errors.
- `MasterJourney`: Tests for the **Master** page that check the search, sorting, filtering and grouping features built into the app.
- `FLPIntegrationJourney`: This journey is available if you have enabled SAP Fiori launchpad for your app. It tests the SAP Fiori launchpad integration features **Save as tile** and **Share on SAP Jam**.
- `AllJourneys`: This is a convenience journey that will call all the other journeys specified above and is used in the test suite file.

You can execute all journeys by calling the test suite file `opaTests.qunit.html` or `opaTestsPhone.qunit.html` in the `webapp/test/integration` folder or selecting the **run all integration tests** link in the `test.html` file in the app's root folder.

For more information, see Integration Testing with One Page Acceptance Tests (OPA5) and sap.ui.test.Opa5 in the **Samples** within the Demo Kit.

Unit Tests

In the `unit` subfolder you can find all unit tests for our application. They are structured similarly to the structure of the `webapp` folder. For example, controller tests are located in the `controller` folder whereas formatter tests are located in the `model` folder.

Unit tests are included for the following functionality:

- ListSelector tests
- Formatters
- Device model

As with the integration tests, you can execute all unit tests by calling the test suite file `unitTests.qunit.html` in the `webapp/test/unit` folder or selecting the **run all unit tests** link in the `test.html` file in the app's root folder.

For more information, see Unit Testing with QUnit, https://qunitjs.com/ and http://sinonjs.org/ .

Device Adaptation

The following outlines the best practices for ensuring your list-detail apps adapt to different kinds of devices in the best way possible.

Content Density

The app templates include a mechanism to adjust the content density of the controls according to the device features. On devices that feature touch support, the controls are automatically displayed larger. For more information, see How to Use Densities for Controls.

Stable IDs

Setting stable IDs is crucial if your app is used in combination with certain functions.

Most controls in the template apps (except for aggregations that are created dynamically, such as list items) are assigned a stable ID to identify the controls in integration tests, extensibility tools like key user adaptation, as well as interactive inline help tools.

Related Information

SAPUI5 Flexibility: Adapting UIs Made Easy
Extending Apps
Stable IDs: All You Need to Know

Worklist Template

The **SAP Fiori Worklist Application** template implements a typical worklist template, one of the patterns that is specified by the SAP Fiori design guidelines.

A worklist displays a collection of items to be processed by the user and usually involves reviewing details of a list item and taking action. If the data needs to be organized into columns or the overview of the items is more important than showing the item details directly, this template can be used as a starting point. For more information about worklist templates, see the SAP Fiori Design Guidelines.

ℹ Note

You have two options: You can use this template to build an **app for the SAP Fiori launchpad** or to build **standalone apps**.

- If the app runs in SAP Fiori launchpad it also contains additional features like **Save as Tile** or **Share in SAP Jam** that depend on SAP Fiori launchpad at runtime. This app cannot be run standalone, meaning no `index.html` file is created but only files for testing the app in the SAP Fiori launchpad sandbox.
- Only standalone apps contain an `index.html` file that is used to start the app.

The **Worklist** view is the main view that is initially displayed in this app. When a user clicks or taps an item in the table, the **Object** view is displayed, showing more details for the selected item. We use the semantic `FullscreenPage` control as the page for both. A `SemanticPage` is an enhanced `sap.m.Page` that contains controls with a semantic meaning and displays them according to the SAP Fiori Design Guidelines, for example. For more details about semantic controls, see the sample in the Demo Kit.

The table in the **Worklist** view displays a header area that shows the current amount of items in the worklist and a search field. The number of items are updated automatically and the search filters for a preconfigured column of the table.

ℹ Note

As the use cases for apps using a worklist pattern differ greatly, we only show a basic scenario in our template as a starting point for your individual development activities. For more information, see How Do I Enhance the Template?

Where Can I Find the Worklist Template?

You can find the template in the following places:

- **SAP Fiori Worklist Application** (for OData V2 models) and **SAP Fiori Worklist Application - OData V4** (for OData V4 models) templates in SAP Fiori tools
- `openui5-worklist-app` in the SAP Repository on GitHub

  For more information on how to clone or download the template from GitHub, refer to the template documentation on GitHub .

Tutorial

See the Worklist App tutorial for an example of how this application can be extended. The result of this tutorial can be seen as the **Manage Products** app in the **Demo Apps** section of the Demo Kit.

How Do I Enhance the Template?

In our template, we use a simple layout that you can use as a basis for enhancements. For example, if you want to use an object page with a dynamic header, you can use one of the page-type Object Page Layout samples in the Demo Kit. All you have to do is replace the relevant content in the template with the content from the sample.

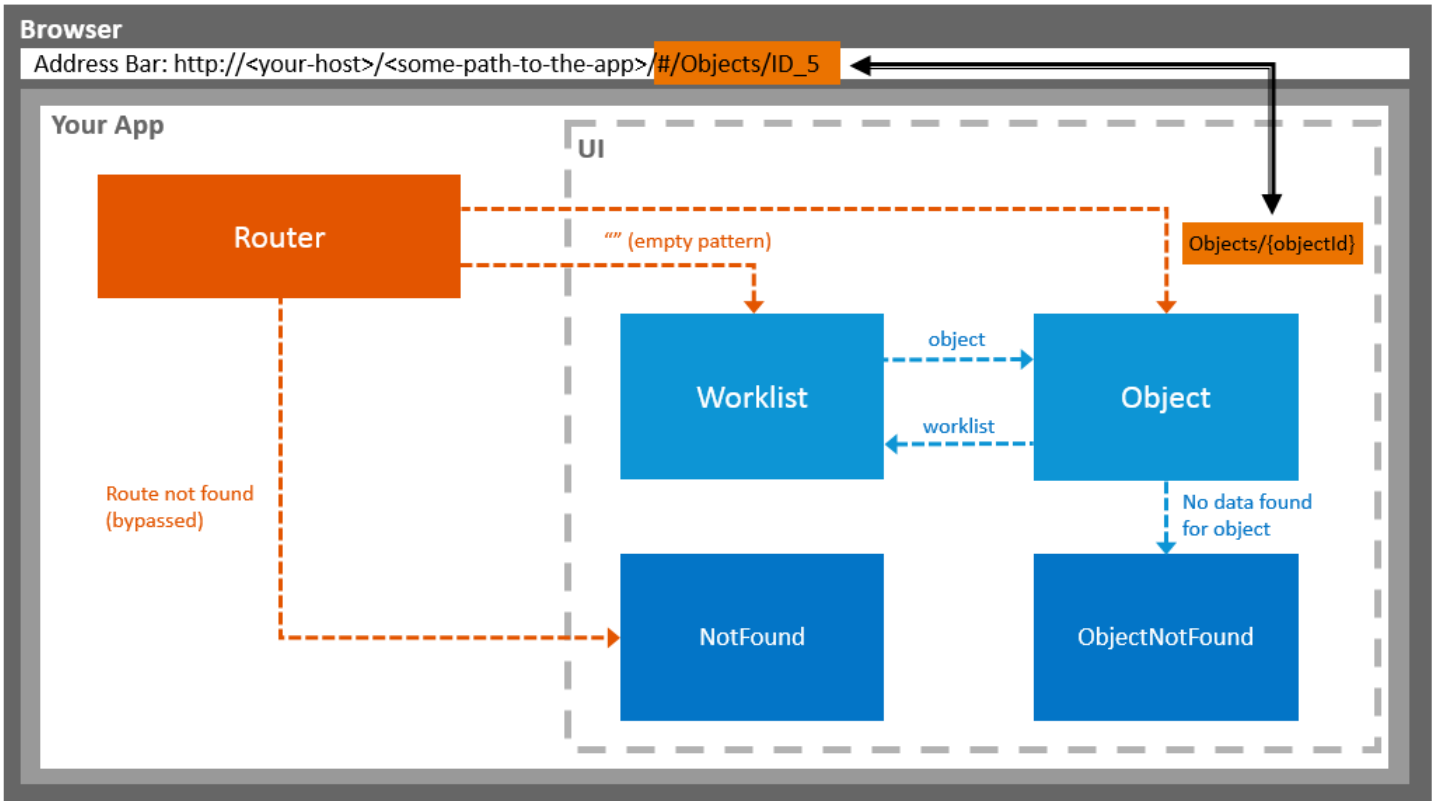
You can find more information about the possibilities of object pages at SAP Fiori Design Guidelines - Object Page.

Related Information

Demo Apps
Development Environment
Worklist App

Navigation

The navigation flow of the Worklist application is very simple as it only contains two main views and the **not found** pages that are displayed as a message to the user in case of navigation errors.



Navigation Flow of the Worklist App

The two main views **Worklist** and **Object** each have a route and a target configured. When the route matches the URL, the target is displayed and the corresponding view is created. For more information, see Routing and Navigation.

Here is a sample implementation for navigating from the worklist to the object page. First you have to implement a press handler on the `ListItem`. Inside, you extract the current ID of the object pressed by the user by using its `bindingContext`. Since we want to navigate to the "object" route, you need to supply the mandatory `objectId` parameter and pass it to the `navTo` function, as described in the sap.ui.core.routing.Routing#navTo section of the **API Reference** in the Demo Kit and shown here:

```
/**
 * Event handler when a table item gets pressed
 * @param {sap.ui.base.Event} oEvent the table selectionChange event
 * @public
 */
onPress : function (oEvent) {
   // The source is the list item that got pressed
   this.getRouter().navTo("object", {
      objectId: oEvent.getSource().getBindingContext().getProperty("ObjectID")
   });
},
// more controller code
```

After calling `navTo`, the hash of the browser is updated and you get an event on the `ObjectController` when the route "object" matches the current hash. In the event handler, you extract the `objectId` using the `Event.getParameter` function. You then bind the data to the view:

```
// init function of the object controller
onInit : function () {
   var oView = this.getView();
   var oModel = oView.getModel();
   this.getRouter().getRoute("object").attachPatternMatched(function (oEvent) {
      var sObjectId =  oEvent.getParameter("arguments").objectId;
      oModel.metadataLoaded().then(function() {
         var sObjectPath = oModel.createKey("Objects", {
            ObjectID :  sObjectId
         });
         oView.bindElement({
```

```
                        path: ("/" + sObjectPath)
                });
        });
    });
        …
    // more init code
},
…
// more controller code
```

notFound (similar to an HTTP 404 "not found" status code)

The **not found** pages are implemented using an `sap.m.MessagePage` an `sap.m.MessagePage`. They display an error message according to the SAP Fiori UX specifications. There are different "not found" cases that each have a separate target and a **notFound** view.

If you have the following URL, no route will match: `index.html/#/thisIsInvalid`. This means that the **notFound** view will be displayed, as the target **notFound** is defined in the bypassed section.

The code sample below shows the relevant parts of the configuration. For a full implementation of a **not found** page, see Step 3:Catch Invalid Hashes.

```
"routing": {
        "config": {
                …
                "bypassed": {
                        "target": "notFound"
                }
        }
        …
  "targets": {
        …
        "notFound": {
                "viewName": "NotFound",
                "viewId": "notFound"
        }

}
```

objectNotFound

If the object route matches – an ID is passed (for example `#/Objects/1337`) but the back end does not contain an object with the ID `1337`, then you need to display the **objectNotFound** page. This is achieved by listening to the "change" event of a binding. Inside this, you check if there is no data and tell the router to display the **objectNotFound** target, as shown in the sample code below:

```
// inside of a controller
this.getView().bindElement({
        path: "/Objects/1337",
   change: function () {
       // there is no data
       if (!this.getView().getElementBinding().getBoundContext()) {
                this.getRouter().getTargets().display("objectNotFound");
                                        return;
                                }
           // code handling the case if there is data in the backend
      …
```

```
    };
});
```

The routing configuration for this navigation flow is set up in the descriptor for applications (`manifest.json` file), as shown here:

```
"routing": {
  "config": {
        "routerClass": "sap.m.routing.Router",
        "viewType": "XML",
        "viewPath": "sap.ui.demo.worklist.view",
        "controlId": "app",
        "controlAggregation": "pages",
        "bypassed": {
          "target": "notFound"
        }
  },
  "routes": [
        {
          "pattern": "",
          "name": "worklist",
          "target": "worklist"
        },
        {
          "pattern": "Objects/{objectId}",
          "name": "object",
          "target": "object"
        }
  ],
  "targets": {
          "worklist": {
                "viewName": "Worklist",
                "viewId": "worklist",
                "viewLevel": 1
          },
          "object": {
                "viewName": "Object",
                "viewId": "object",
                "viewLevel": 2
          },
          "objectNotFound": {
                "viewName": "ObjectNotFound",
                "viewId": "objectNotFound"
          },
          "notFound": {
                "viewName": "NotFound",
                "viewId": "notFound"
        }
  }
}
```

For more information, see Routing and Navigation, the `sap.m.routing.Router` section of the **API Reference** documentation in the Demo Kit, and the `sap.ui.core.routing.Router` sample within the Demo Kit.

Busy Indication

The Worklist app implements a busy indication concept as specified by the SAP Fiori Design Guidelines.

Calling the app will result in the following:

- Only initially a global busy indicator is displayed that overlays the whole app until the metadata of the service is loaded.
- A local busy indicator is displayed on the worklist table or on the page of the object view while the data from the service is loading.

- When controls are loading additional data or getting refreshed, a local busy indication is displayed automatically.

By default, the busy indicator delay is set to one second for all controls. This would first show the UI for a second, then show a busy indication until the data is loaded. To avoid this behavior initially and show the busy indicator immediately without delay the following concept is implemented in the app: The `busyIndicatorDelay` and `busy` properties of certain controls (`AppView`, `Table` on the **Worklist** page, `FullScreenPage` on the **Object** page) are bound to the local view model and manipulated in the controllers of the app. The delay is initially set to "0" for displaying the busy indicator immediately, and reset to the previous value after the initial loading is done.

ℹ Note

You can find more information about busy indicators, busy states, and busy handling in general in the SAP Fiori Design Guidelines.

Model Instantiation

The application configures several data models that are used throughout to update the views or to store additional configuration options.

The service model and the resource bundle are instantiated automatically by the app's component during startup and described in the first section. The local view models and helper models such as the device model are set up as JSON models and described in the second section.

Automatic Model Instantiation

The templates instantiate the service and resource model automatically using the following configuration entries in the descriptor. When the component of the app is initialized, these models will be made available under the configured name throughout the application.

An external service is defined in the `dataSources` section of the `sap.app` namespace. In the example shown below, we configure an OData V2 model and the alias `"mainService"` in the `manifest.json` descriptor file:

```
{
        …
        "sap.app": {
                …
                "i18n": "i18n/i18n.properties",
                …
                "dataSources": {
                        "mainService": {
                                "uri": "/here/goes/your/serviceUrl/"
                                "type": "OData",
                                "settings": {
                                        "odataVersion": "2.0",
                                        "localUri": "localService/metadata.xml"
                                }
                        }
                },
                …
        },
        …
}
```

ℹ Note

If you use the OData V4 template, you set the `odataVersion` accordingly.

In the models section of the `sap.ui5` namespace we define two models that will be instantiated automatically. The resource model is a named model (i18n) and the OData model is the default model so it has no name. The OData model also receives additional URL parameters via the `metadataUrlParams`. The parameters `sap-server`, `sap-client`, and `sap-language` are passed to the service automatically by SAPUI5, as shown in the following `manifest.json` code snippet:

```
{
        …
        "sap.ui5": {
                …
                "models": {
                        "i18n": {
                                "type": "sap.ui.model.resource.ResourceModel",
                                "settings": {
                                        "bundleName": "sap.ui.demo.masterdetail.i18n.i18n",
                                        "supportedLocales": [""],
                                        "fallbackLocale": ""
                                }
                        },
                        "": {
                                "dataSource": "mainService",
                                "preload": true
                        }
                },
                …
        }
}
```

ℹ Note

Before SAPUI5 version 1.30, all models were defined and instantiated in the component's `init` method. We recommend removing all manual model creation code and switching to the automatic model instantiation instead. The "device model" however is still a local model that has to be instantiated manually.

Additional Models for the App

The following models are created as local JSON models in the application and can be referenced by its model name where needed:

- **device**

  The device model provides an easy access to the `sap.ui.Device` API and is used to configure certain view settings according to the user's device.

- **FLP**

  The SAP Fiori launchpad model is a helper module to configure SAP Fiori launchpad integration and is used to control the sharing options of the application.

- **worklistView**

  A local view model for the worklist view that stored configuration options that are bound to controls in the view.

- **objectView**

A local view model for the object view that stored configuration options that are bound to controls in the view.

Send Email

The **Send Email** feature is a sharing option that can be found in the share menu of each view.

This feature simply triggers a `sap.m.URLHelper` action that will show a new email with preconfigured texts in the default client of the user.

The placeholder texts are located in the resource model and can be adjusted to your use case. The texts already include the current context and the current location. The texts may vary for each view, therefore they are configured with the local view model and updated when the business object context has changed.

For more information about `sap.m.URLHelper`, see the sample in the Demo Kit.

Testing

The templates include basic testing features, unit tests as well as integration tests for a basic test coverage of the initial app. The tests are written independently of the actual data displayed in the app.

The `webapp` folder of the template app contains a `test.html` file which serves as an overview for the different test pages. You can run the app with or without mock data and run the unit and integration tests. This section describes which application tests are provided and how they are structured.

Integration Tests

The integration tests shipped with the template cover all basic functionality and provide several "journeys". Journeys include a series of OPA tests that belong to the same functionality and should be executed together:

- `NavigationJourney`: This journey will trigger user interactions and navigate through the application. The routing configuration, basic navigation events, and error handling are tested here.
- `NotFoundJourney`: Several "not found" cases of the application are tested here. Faulty navigation scenarios are introduced intentionally to simulate errors.
- `WorklistJourney`: A series of tests for the **Worklist** page that check the busy indication and sharing features built into the app.
- `ObjectJourney`: A series of tests for the **Object** page that check the busy indication and sharing features built into the app.
- `FLPIntegrationJourney`: This journey is available if you have enabled SAP Fiori launchpad (FLP) for your app. It tests the FLP integration features **Save as tile** and **Share on SAP Jam**.
- `AllJourneys`: This is a convenience journey that will call all the other journeys specified above and is used in the test suite file.

You can execute all journeys by calling the test suite file `opaTests.qunit.html` in the `webapp/test/integration` folder or selecting the **run all integration tests** link in the `test.html` file in the app's root folder.

For more information, see Integration Testing with One Page Acceptnce Tests (OPA5) and sap.ui.test.Opa5 in the **Samples** within the Demo Kit.

Unit Tests

In the `unit` subfolder you can find all unit tests for our application. They are structured similarly to the structure of the `webapp` folder. For example, controller tests are located in the `controller` folder whereas formatter tests are located in the `model` folder.

Unit tests are included for the following functionality:

- App controller tests
- Worklist controller tests
- Formatters
- Device model

As with the integration tests, you can execute all unit tests by calling the test suite file `unitTests.qunit.html` in the `webapp/test/unit` folder or selecting the **run all unit tests** link in the `test.html` file in the app's root folder.

For more information, see Unit Testing with QUnit,, https://qunitjs.com/ and http://sinonjs.org/ .

Device Adaptation

The following section outlines the best practices for ensuring your worklist apps adapt to different kinds of devices in the best way possible.

Content Density

The app templates include a mechanism to adjust the content density of the controls according to the device features. On devices that feature touch support, the controls are automatically displayed larger. For more information, see How to Use Densities for Controls .

Stable IDs

Setting stable IDs is crucial if your app is used in combination with certain functions.

Most controls in the template apps (except for aggregations that are created dynamically, such as list items) are assigned a stable ID to identify the controls in integration tests, extensibility tools like key user adaptation, as well as interactive inline help tools (such as Web Assistant 2, formerly known as xRay).

Related Information

SAPUI5 Flexibility: Adapting UIs Made Easy
Extending Apps
Stable IDs: All You Need to Know

Data Source

This section provides information on how to connect your application with a data source during generation. In the Data Source and Service Selection wizard page, you can select from one of the following options:

Connect to an SAP system

- **Connect to an SAP system using VS Code**

  In either case, you can create a system to connect to or select from any saved systems you may have already used.

  Adding a new system

  Enter a system name to identify the new system that will be unique to use in SAP Fiori application generator or in the **SAP Systems View**.

  Select a Service Key. For an SAP ABAP system hosted in the SAP Business Technology Platform, you must provide a service key that contains the key information for the required SAP ABAP system. This service key should be provided by your administrator for the selected SAP ABAP system. Once this information is provided, a browser tab launches and prompts you to authenticate against the system.

  For more information on how to create a service key, please see Create Service Keys Using the Cockpit.

  ### ℹ Note

  Once you have authenticated your user for the new system in the browser, SAP Fiori application generator displays you the list of OData services available for the user you have used to log in. Please see the **Service** dropdown with the title **Service (for user [<USERNAME>])**.

  For an on-premise SAP ABAP system, you need to provide the system URL and optional client ID, along with the authentication details for that system if required.

  ### Example

  https://ldciu1y.wdf.sap.corp:44355, client: 010

  In both scenarios, you can store the system details in the secure storage of your operating system.

  - Microsoft Windows: Keychain.
  - MacOS: Credential Manager.

  Saving the system in this way ensures that you do not need to continually provide these details for generating an application or running the generated application locally.

  A saved system can be deleted from either of these places, as needed.

  ### ℹ Note

  When connecting for the first time, only the **New System** option is available.

- **Connecting to an SAP System using SAP Business Application Studio.**

  When using the SAP Fiori application generator in SAP Business Application Studio, you can select from a list of destinations that are configured for **Business Application Studio** instance. The generator automatically retrieves the available destinations, and you can select from the list. If you do not have the correct access to use the destination end point, an error occurs.

Connect to an OData Service with a Customized URL

If the OData endpoint that you want to use in your application can't be accessed directly, you can set it up as a destination and directly reference it in the generator. To do so, perform the following steps:

1. In SAP Business Application Studio, launch the SAP Fiori generator and select the required template.
2. Select **Connect to an OData Service** from the data source drop-down list.

3. For the data source URL field, use the destination name followed by `.dest`. In this case, SAP Business Application Studio should be able to route to your service with the destination name.

Example

If the URL defined in the Destination is `https://someurl.com/someservice` with the destination name "MyDestination", the following URL will be used in the SAP Fiori generator:

```
https://MyDestination.dest/someservice
```

Connect to an OData Service

Enter the OData endpoint URL to generate your application. All OData endpoints that are either **authenticated** with Basic authentication or **unauthenticated** are supported.

Note

The provided OData endpoint must be the correct version for the template that you select. For example, a V2 endpoint must be provided for the V2 template. The wizard informs you if there is any mismatch between the OData version and the template version.

If necessary, the system prompts you to provide your name and password.

Upload a Metadata Document

To generate the application without relying on a backend service being available, upload a metadata xml file that you want to use.

Only EDMX format is supported for metadata xml file.

Once the metadata xml file has been validated, the system allows you to select the required entity options for the application.

Note

When using a `metadata.xml` file, the generated application is limited to only mock data.

Connect to SAP API Business Hub

When users do not have their data source available, they can generate an application with the SAP API Business Hub. This data source is only intended to support the development and should be replaced with a real one before going live. When the SAP API Business Hub option is selected, a list of predefined services relevant to different industries appears.

- Select a service that you want to generate an application with. For example, Just-In-Time Calls, Transaction Classifications, Content, Request of Quotation, and more.
- Once the service is selected, two more fields appear for authentication purposes: **Enter your Username** and **Enter your Password**.
- Fill in the fields and click **Next** to proceed with the application generation.

Note

You cannot deploy applications that use the SAP API Business Hub, as this data source is intended for local development only.

For more information, see https://api.sap.com/.

Use a Local CAP Project

You can select a local SAP Cloud Application Programming Model (CAP) project in your filesystem and the generator retrieves the services that are defined for that project. The folder location you provide is validated to ensure it is an SAP Cloud Application Programming Model (CAP) Node.js project that the generator can support.

For more information about CAP services, see: https://cap.cloud.sap/docs/about/.

Template Properties

Once the data source is supplied, you can customize the application by selecting from the list of entities in the data source.

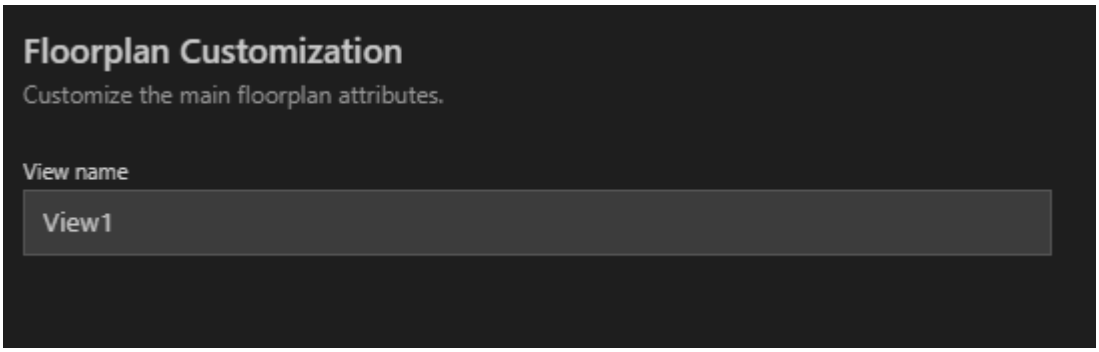
Generic options across multiple template types:

- **Object Collection**. The OData entity set used to display the data in the selected template.
- **Object Collection Key**. The key property of the entity set, which makes the entry unique.
- **Object ID**. The display name of the entity to be used.
- **Object Number**. The number field displayed on the right side of the table data.
- **Object Unit of Measure**. The measurement unit for the Object number above.

Select the entity set to be used from the drop-down list of entities. Note, that this is a mandatory field. After selecting the entity set, a list of properties appears that is filtered depending on the selected entity.

The **SAP Fiori List-Detail Application** specific template option is a list of line items for the detail page, with similar meaning as object page values.

The **Basic** application template has an additional property, which is the **View name**. This is a unique name selected for the SAPUI5 view to be created.



Additional Configuration

Add Deployment Configuration.

1. Select **Yes** if you want to configure deployment settings.
2. From the **Choose the target** drop-down list, select the deploy target from the following options:
    - ABAP
    - Cloud Foundry

When the target system is selected, the SAP Fiori launchpad wizard steps with the prompts appear **Deploy Configuration**.

ABAP system

- **Destination name** (required). The ABAP destination to the back-end system.
- **Is this an SAP Cloud Platform system?**

  Select **Yes** or **No**. Applicable only if the system is discovered to be a Steampunk system.
- **Target System URL**. This field has already been prefilled according to the selected project.

  Not applicable in SAP Business Application Studio.
- **Client**. A self-contained commercial, organizational, and technical unit within an SAP system. Business data of clients are separated from each other, and clients serve a specific purpose in an SAP instance.

  Not applicable to a Steampunk system and in an SAP Business Application Studio.

  The following options are available to select from:
    - Use project defined client - xxx
    - Enter client
    - Use default system client
- **Is this an SAP S/4HANA Cloud system?** Select **Yes** or **No**. Not applicable if the user responds **Yes** to the *Is this an SAP Cloud Platform system?* question.
- **SAPUI5 ABAP Repository** (required). The technical name of the SAP Fiori application being deployed.
- **Package**. A transportable ABAP repository object that groups development objects. An SAP Fiori application deployed in the ABAP backend must be assigned to an appropriate package. If it's supposed to be a local application that isn't sent to another system, the package can be $TMP. In this case, there's no Transport Request. Not applicable for SAP S/4HANA Cloud.

- **Fetch Transport Request list from target system?**. Select **Yes** or **No**. If you select **Yes**, the applicable list of transport requests is retrieved from the target system and displayed in a dropdown for you to choose. If the lists of transport requests are unable to be retrieved from the target system, the existing manual text entry will be required.
- **Transport Request number**. The ID of a container that is used to transfer data from one SAP installation to another. It collects changes made in a development system, allowing distribution of them across the defined transport landscape.

  Not applicable for SAP S/4HANA Cloud system. For more information on transport requests, see Managing Transport Requests.

Cloud Foundry system

- **Destination name** (required). The Cloud Foundry destination to the back-end system.

Add FLP Configuration

Select **Yes** if you want to add an SAP Fiori Launchpad configuration. A new step appears with the **FLP Configuration** prompts.

> **Note**
> If the configuration already exists, the existing values are displayed. In this case, the user still can change the inputs.

- **Semantic Object**. Represents a business entity, such as a customer, a sales order, or a product. Using semantic objects, the user can bundle applications that reflect a specific scenario and refer to objects in a standardized way, abstracting from concrete implementations of these objects. It's used in mapping URLs of SAP Fiori applications to objects in the launchpad.
- **Action**. Describes which operation, such as display or approve Purchase Orders, is intended to be performed on a semantic object. For example, Purchase Order or Product.
- **Title**. The name of the Fiori application that appears on the SAP Fiori launchpad tile in a free text format.
- **Subtitle** (optional). A free text field where the user can further describe the application in the launchpad.

Configure Advanced Options

Select **Yes** if you want to configure advanced options.

- Select SAPUI5 theme:
  - Quartz Light
  - Belize
  - Quartz Dark

    > **Note**
    > Quartz Dark is only available on SAPUI5 versions 1.72 and later.

  - Morning Horizon
  - Evening Horizon

    > **Note**
    > Morning Horizon and Evening Horizon are shown in the dropdown if the minimum SAPUI5 version selected is **1.102** or above. **Morning Horizon** is selected by default.

- **Add Eslint configuration to the project**.

Choosing this option includes the Fiori eslint plugin library in the generated application that allows the developer to check and ensure that the Fiori application adheres to the best practice for Fiori code development. Executing the target `npm run lint` in the generated application checks for any linting errors.

- **Add javascript code assist libraries to your project**

  With this option, libraries are included in the generated project to provide `ui5` code completion prompts in the editor along with the `eslint` rules and recommended configuration for the static `jsdoc` code checks.

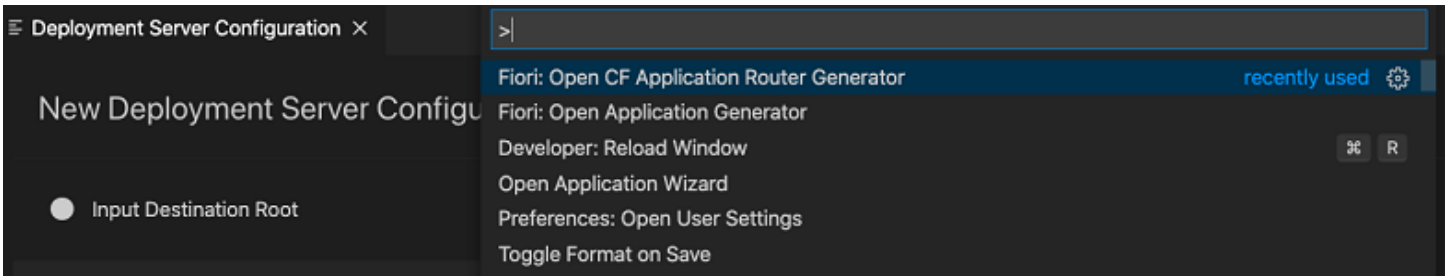
  For more information, see Add Javascript Code Assist.

- **Enable TypeScript (Experimental)**

  You can optionally choose to generate your application with **TypeScript** support. This is currently an experimental feature and is subject to future updates to enhance support.

Generate an MTA Deployment File

In the MTA deployment scenario, developers can generate an app router configuration, that contains the `mta.yaml` file, and then add multiple generated SAP Fiori apps to the app router configuration project. To do so, use the command *Fiori: Open CF Application Router Generator*.



In this case, the user can manage the source code of the app router configuration and multiple SAP Fiori elements projects under a single root directory.

The app router configuration project has the following structure:

- `router`. A folder that contains app router configuration.

  ⓘ Note

  This folder can have a different name, such as `configurable`.

- `.gitignore`.
- `mta.yaml`. The configuration `mta.yaml` file.
- `package-lock.json`. The file is generated automatically for any operations where npm modifies either the `node_modules` tree or `package.json` and describes the exact tree that was generated.
- `package.json`. Contains specifics of the `npm package.json` handling.

Once the app router configuration project is generated, one or more SAP Fiori apps can be generated inside its root directory by using the SAP Fiori application generator.

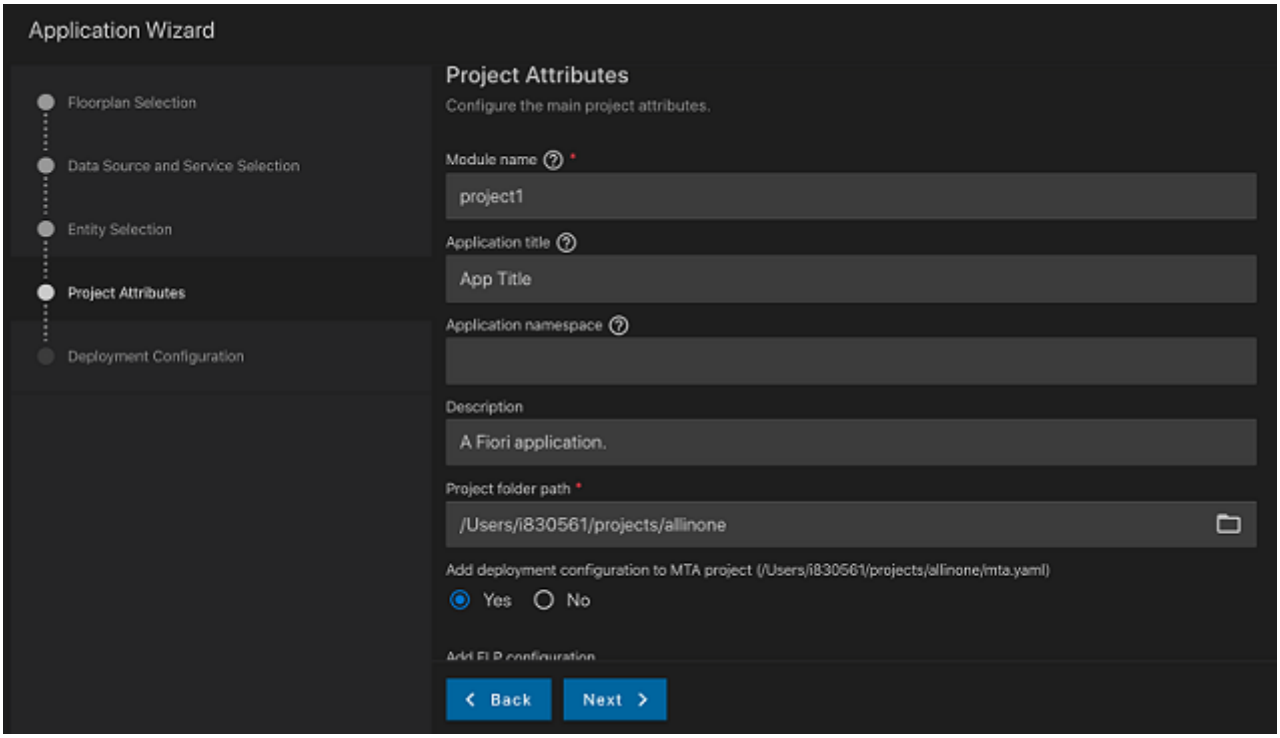
ⓘ Note

There are two types of routers standalone vs. managed. To see the differences please see a blog: SAP Tech Bytes: FAQ Managed Approuter vs. Standalone Approuter or Developing HTML5 Applications in the Cloud Foundry Environment.

Add a Fiori Application to an MTA Deployment File with the SAP Fiori Application Generator

Add Deployment Configuration to an Existing MTA Deployment File

When the user selects a project folder path for an application and an MTA file already exists in the selected folder, the SAP Fiori application generator automatically switches to **Yes** for adding deployment configuration. During the generation of an application, the generator updates the MTA project file by default with the supplied deployment configuration.



ℹ Note

For successful deployment of the generated application, the MTA file must already have the required services defined, depending on whether the application has to be deployed by using the Standalone or Managed application router. To ensure the required services are added to the MTA deployment file, use the MTA generator provided by SAP Fiori tools or the Application Router generator in SAP Business Application Studio. For more information, see Generate an MTA Deployment File.

ℹ Note

For local CAP projects, you can optionally chose to generate an instance based destination as part of CF deployment configuration, or use an existing instance based destination that is already defined in your MTA deployment file.

Create an MTA Deployment File during Application Generation

If the MTA deployment file is not available in the project folder location, you can create a new file. To do so, when adding deployment configuration, select **Cloud Foundry** as the target landscape. In this case, the relevant MTA file to be used with a managed application router will be added to the same folder as your generated SAP Fiori application.

ℹ Note

In this scenario, the generated MTA file is not applicable for use with multiple applications, as it is contained in the same folder as the SAP Fiori application. We recommend that you first create the MTA file and then generate the SAP Fiori application in a subfolder.

Add Javascript Code Assist

You can modify an SAP Fiori project with javascript code assist libraries in the **Advanced configuration** step.

To add javascript code assist libraries to the already generated application with SAPUI5 version 1.76 and newer, perform the following steps in your project:

1. Update the `package.json` file:

```
"devDependencies": { "eslint": "5.16.x", "@sap/eslint-plugin-ui5-jsdocs": "2.0.x", "@sapui5/ts
```

2. Add `tsconfig.json` with the following content to the root folder :

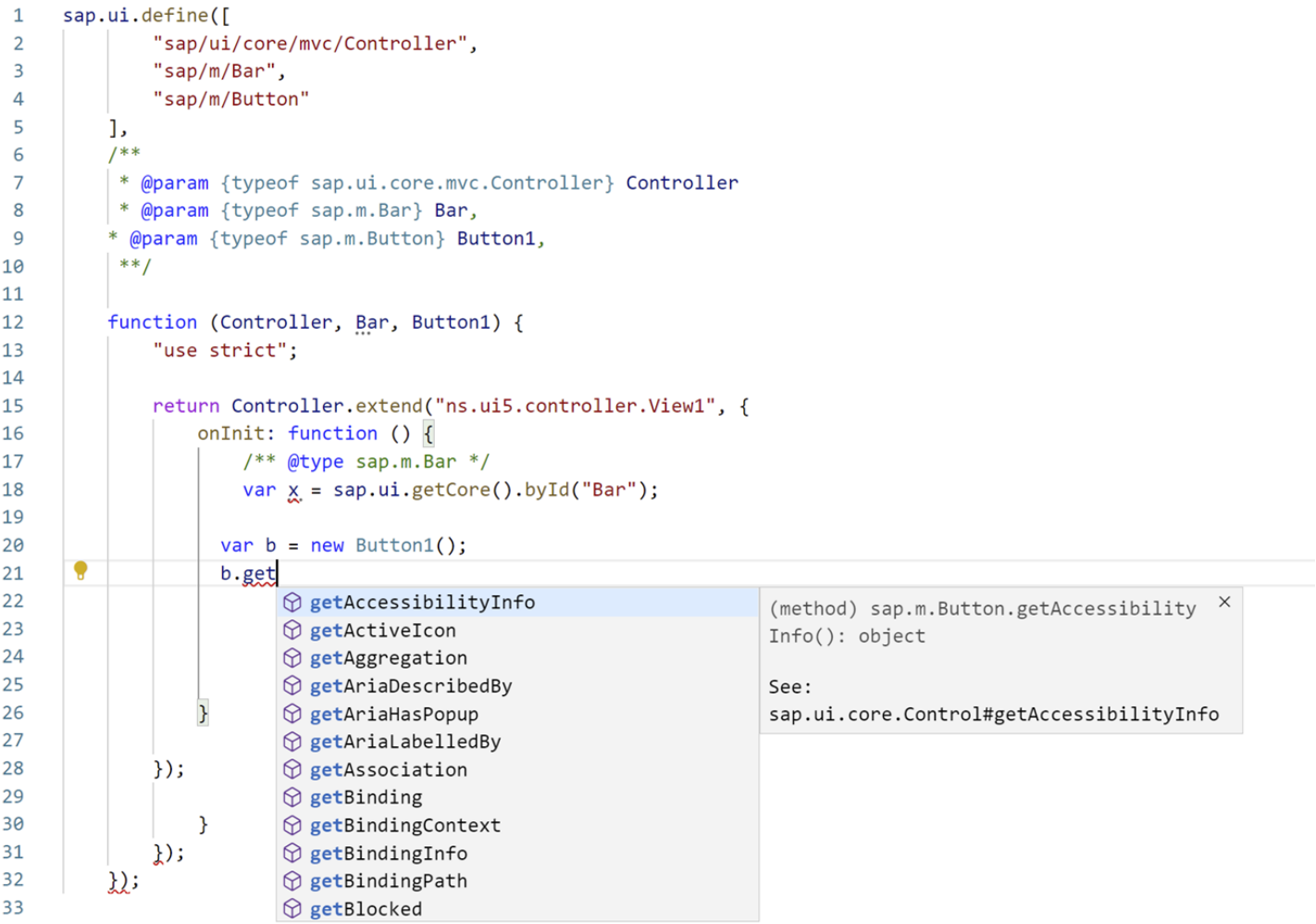
```
{ "compilerOptions": { "module": "none", "noEmit": true, "checkJs": true, "allowJs": true, "ty
```

3. Add the `.eslintrc` file with the following content to the root folder:

```
{ "plugins": ["@sap/ui5-jsdocs"], "extends": ["plugin:@sap/ui5-jsdocs/recommended", "eslint:r
```

4. Delete `node_modules` and execute `npm install`.

Open a JS file to see the code completion for SAPUI5:



Security Certificate

During application generation, invalid security certificate errors may occur when the system the user connected to is using SSL to support secure HTTPS traffic. In some cases, the certificate is generated using a local certificate authority that is unknown to the user operating system. If this happens, the SAP Fiori application generator rejects the connection request and reports an error.

1. Ensure that you have a copy of the local certificate authority saved locally.

   > **Note**
   >
   > Create a folder or select a folder that can permanently keep your downloaded certificate. To export the certificate from your browser, follow the instructions based on your web browser and its version.

2. Once the certificate file is downloaded, the issue can be resolved for VS Code by importing it into your global certificate store. To do so, perform the following steps:

   - Microsoft Windows.
     - Right-click the **CA certificate** file and select **Install Certificate**.
     - Follow the prompts to add the certificate to the trust store either for the current user only or for all users logging onto this computer.
   - MacOs.
     - Right-click the **CA certificate** file.
     - Select **Open With** and naviagte to **Keychain Access**.
     - Select **System** as the keychain to import into.

3. To resolve the issue for using the command line with Yeoman, set the following environment variable to point to the location of the downloaded certificate file: `NODE_EXTRA_CA_CERTS`

   - Microsoft Windows:
     - Right-click the **Computer** icon and select **Properties**. Alternatively, in Windows Control Panel, select **System**.
       - Microsoft Windows 10
         - Right-click **Windows Start Button** and select **System**.
         - In **System Settings** under **Related Settings**, select **System info**.
       - Select **Advanced system settings**.
       - On the Advanced tab, select **Environment Variables**.
       - Click **New** to create a new environment variable.
       - Add `NODE_EXTRA_CA_CERTS` and ensure the value points to the downloaded certificate file that is stored in the folder from **Step 1**.
       - Click **Apply** and then **OK** for the changes to take effect.

     MacOs

     - In the command-line terminal, before executing the Yeoman command to run the generators, execute the following command:

       ```
       export NODE_EXTRA_CA_CERTS=path/to/certificate/file
       ```

Replace `path/to/certificate/file` with the location of the downloaded certificate.

Preview an Application

Note

In the development environment, localhost is used as an HTTP proxy server to reach the backend OData service. A domain security policy, that is applied in some companies, includes localhost which forces HTTPS to be used in redirected URLs. As a result, when the user attempts to access a website, an error may occur.

If an SSL protocol error appears after the URL redirection, check the security configuration of your browser. For example, Chrome HSTS configuration can be accessed by entering the following URL address to the browser: `chrome://net-internals/#hsts`.

Start a Preview from Terminal

After the application is successfully generated, several options to preview it are available by running `npm start` scripts in the terminal:

- Use Live Data
- Use Mock Data
- Use Local Sources
- Use Custom Middlewares

Start a Preview with Run Control

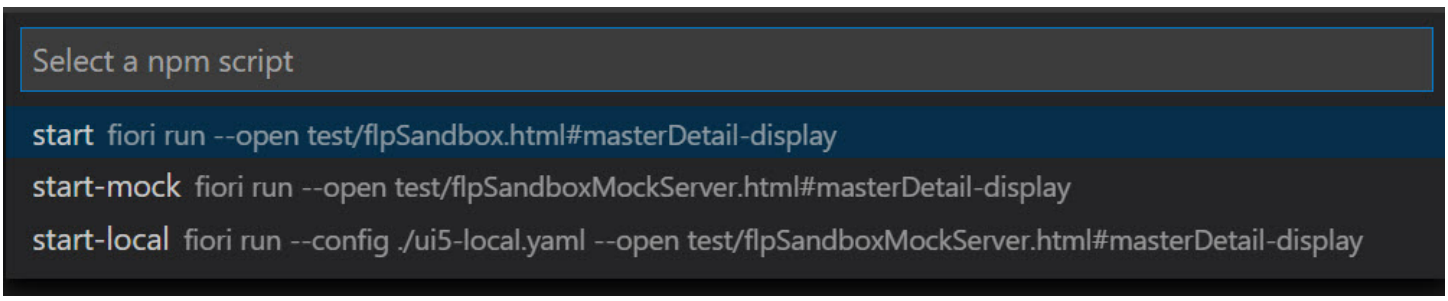
For various options of starting your application, use the **Run Control** function in VS Code or SAP Business Application Studio. Also, you can create a new *Run Configuration* in SAP Business Application Studio and *Launch Configuration* in VS Code:

- Use Run Control
- Create a New Run Configuration in Visual Studio Code
- Create a New Run Configuration in SAP Business Application Studio

Start a Preview from Context Menu

This section provides instructions on previewing an application from a context menu in VS Code and SAP Business Application Studio.

Alternatively to the execution of the start scripts on the command line, right-click the project folder or any subfolder and select **Preview Application**. You are then provided with three options:



1. **start**. Starts the application using `npm start`. Runs the application using the live service to retrieve real data against the hosted version of SAPUI5 that was selected during generation. Then, the application runs on `localhost:8080` and connects to the live OData service endpoint. If the OData endpoint requires authentication, the browser prompts you to enter your credentials

2. **start-mock**. Starts the application using `npm run start-mock`. Runs the application with mock data against the hosted version of SAPUI5 that was selected during generation. Then, the application runs on `localhost:8080` but uses a mock server to reflect the OData endpoint. This way, you can use the application without connecting to a live OData service.

3. **start-local**. Starts the application using `npm run start-local`. Runs the application with mock data against a local copy of the SAPUI5 library that was selected during generation. Then, the application runs on `localhost:8080` but uses a mock server to reflect the OData endpoint. This way, you can use the application without connecting to a backend.

ℹ Note

The automatic download is only supported with SAPUI5 versions 1.76 and higher.

Enable App-to-App Navigation Preview

- App-to-App Navigation Preview

Preview an SAP Fiori Elements CAP Project

Once your CAP project is generated, you can preview it in Visual Studio Code or SAP Business Application Studio.

Visual Studio Code

To run an application preview in VS Code, perform the following steps:

1. In VS Code, open the terminal.

   ### ℹ Note

   To open the terminal in VS Code:

   - Use the Ctrl+` keyboard shortcut with the backtick character.
   - Use the **View** > **Terminal** menu command.
   - From the **Command Palette** (Ctrl+Shift+P), use the **View: Toggle Integrate Terminal** command.

2. Ensure you are in the root directory of your project.
3. In the terminal, type `cds run` and press **Enter**.

   A new line appears, such as:

   ```
   server listening on { url: 'http://localhost:4004' }
   ```

   ### ℹ Note

   If an error occurs when executing *cds run*, enter the following commands:

   ```
   npm i -g @sap/cds
   npm i -g @sap/cds-dk --force
   ```

4. Open the link in the terminal using the Ctrl+Click combination.

   A new browser window with a list of links opens.

5. Click the HTML link in the list, such as `/incidents/webapp/index.html`.

   Your application is displayed on the launchpad.

6. Navigate back to VS Code and stop the server by executing `Ctrl-C` in the terminal window.

SAP Business Application Studio

To run an application preview in SAP Business Application Studio, perform the following steps:

1. In SAP Business Application Studio, open the terminal.

   Note

   To open the terminal in SAP Business Application Studio:

   - Select Terminal > New Terminal from the menu bar.

2. Ensure you are in the root directory of your project.
3. In the terminal, type `cds run` and press Enter.

   A new line appears, such as:

   ```
   server listening on { url: 'http://localhost:4004' }
   ```

   Note

   If an error occurs when executing *cds run*, enter the following commands:

   ```
   npm i -g @sap/cds
   npm i -g @sap/cds-dk --force
   ```

4. Open the link in the terminal using the Ctrl+Click combination.

   A new browser window with a list of links opens.

5. Click the upper link in the list in HTML format, such as `/incidents/webapp/index.html`.

   Your application is displayed on the launchpad.

6. Navigate back to SAP Business Application Studio and stop the server by executing `Ctrl-C` in the terminal window.

Use Live Data

Start the application using the `npm start` command. Then, the application runs on `localhost:8080` and connects to an OData service endpoint. The preview of the application opens automatically in a new browser tab. If the OData endpoint requires authentication, the browser prompts you to enter your credentials.

VS Code

1. In VS Code, open the terminal.

ℹ Note

To open the terminal in VS Code, perform the following steps:

- Use the Ctrl+` keyboard shortcut with the backtick character.
- Navigate to the View > Terminal menu command.
- From the Command Palette (Ctrl+Shift+P), use the View: Toggle Integrate Terminal command.

2. Ensure you are in the root directory of your project.
3. In the terminal pane, type `npm start` and press Enter.

   The preview of your application starts automatically.

ℹ Note

If `port 8080` is already in use, the system selects the next available port to start the application.

SAP Business Application Studio

1. In SAP Business Application Studio, open the terminal.

ℹ Note

To open the terminal in SAP Business Application Studio, perform the following steps:

- Select Terminal > New Terminal from the menu bar.

2. Ensure you are in the root directory of your project.
3. In the terminal pane, type `npm start` and press Enter.

   The preview of your application starts automatically.

Use Mock Data

Using Mock Data in VS Code

ℹ Note

Mockserver configuration is needed prior to using `npm run start-mock`. See Installing MockServer.

Start the application using `npm run start-mock`. Then, the application runs on `localhost:8080` but uses a mock server to reflect the OData endpoint. This way, you can use the application without having to connect to a live OData service and generate mock data on the fly. If you want to generate .json files for your mock data, see: Data Editor. The preview of the application opens automatically in a new browser tab.

1. In VS Code, open the terminal.

ℹ Note

To open the terminal in VS Code,you can:

- Use the Ctrl+` keyboard shortcut with the backtick character.

- Select View > Terminal in the menu.
- Use the View: Toggle Integrate Terminal command from the Command Palette (Ctrl+Shift+P) .

2. Ensure you're in the root directory of your project.
3. In the terminal pane, type `npm run start-mock` and press Enter.

> **Note**
> If `port 8080` is already in use, the system chooses the next available port to start the application on.

SAP Business Application Studio

> **Note**
> Mockserver configuration is needed prior to using `npm run start-mock`. See Installing MockServer.

Start the application using `npm run start-mock`. Then, the application runs on `localhost:8080` but uses a mock server to reflect the OData endpoint. This way, you can use the application without having to connect to a live OData service.

1. In SAP Business Application Studio, open the terminal.

> **Note**
> To open the terminal in SAP Business Application Studio, you can:
>
> - Select Terminal > New Terminal from the menu.

2. Ensure you're in the root directory of your project.
3. In the terminal pane, type `npm run start-mock` and press Enter.

Installing MockServer

Installing Mock Server

> **Note**
> For a new project that is created with SAP Fiori application generator the mock server configuration is automatically added. See Feature Matrix.

If you've created a project and want to install mock server, you can either:

- Start: *Fiori: Open Application Info*.
  - Under **What you can do** section, click `Add Mock server Config`.
- In the project root, open the terminal, and run `npx @sap-ux/create add mockserver-config` command. On how to open terminal, see Use Mock Data.

> **Note**
> To remove mock server run command `npx @sap-ux/create remove mockserver-config`.

Once the mock server is installed, the following configuration is included in your application:

- `package.json` includes `start-mock` script.
- `package.json` includes `@sap-ux/ui5-middleware-fe-mockserver` as `devDependency` and `UI5` dependency.
- `ui5-mock.yaml` file is included in your project.

You can use the `Add Mock server Config` to update an outdated or missing mock server configuration in your project. This allows you to ensure that your mock server is configured correctly and is up to date. For more information on the available commands, you can run the command `npx @sap-ux/create help` in your project root terminal.

Use Local Sources

Start the application using the `npm run start-local` command. Then, the application runs on `localhost:8080` and uses a mock server to reflect the OData endpoint. A local copy of the SAPUI5 library is downloaded from `npmjs`, if necessary. You can modify the version by updating the `ui5-local.yaml` file in the project folder. This way, you can use the application without connection to a backend.

VS Code

1. In VS Code, open the terminal.

   Note

   To open the terminal VS Code:

   - Use the Ctrl+` keyboard shortcut with the backtick character.
   - Use the View > Terminal menu command.
   - From the Command Palette (Ctrl+Shift+P), use the View: Toggle Integrate Terminal command.

2. Ensure you are in the root directory of your project.
3. In the terminal pane, type `npm run start-local` and press Enter.

   The preview of your application starts automatically.

   Note

   The automatic download is only supported with SAPUI5 versions 1.76 and higher.

SAP Business Application Studio

1. In SAP Business Application Studio, open the terminal.

   Note

   To open the terminal in SAP Business Application Studio:

   - Select Terminal > New Terminal from the menu bar.

2. Ensure you are in the root directory of your project.

3. In the terminal pane, type `npm start-local` and press **Enter**.

   The preview of your application starts automatically.

> **Note**
>
> The automatic download is only supported with SAPUI5 versions 1.76 and higher.

Use Custom Middlewares

You can plug custom middleware implementations into the internal express server of the SAPUI5 Server module if you want to handle requests differently. For example, add various headers to a response or parse data of a POST request in a specific way.

SAP Fiori tools use the capabilities of custom middlewares to start and preview SAP Fiori elements applications:

- Application reload. The middleware to reload the application automatically in the browser, when any change is done.
- Proxy. The middleware to connect to the backend server side and to load the remote SAPUI5 resources.
- Serve static. The middleware to serve local resources from your workspace, such as the reuse libraries.

Application Reload Middleware

With the application reload middleware, developers can preview SAP Fiori elements applications while developing or configuring them. Whenever a file relevant for SAP Fiori elements is changed, the application reload middleware will refresh the application preview.

Example Configuration

To start the application reload middleware with its default settings, execute `npx fiori run` in your project with the configuration below in the `ui5.yaml` file.

```
server:
  customMiddleware:
  - name: fiori-tools-appreload
    afterMiddleware: compression
```

Configuration options

The application reload middleware does not require any configuration parameters. However, there are optional parameters that can be used if the project structure differs from the standard SAP Fiori elements projects.

Configuration parameters

Parameter	Type	Default value	Description
path	`<string>`	`webapp`	Path to be watched. By default, the standard SAPUI5 `webapp` folder is used.
ext	`<string>`	`html, js, json, xml, properties, change`	Change this parameter to select a custom set of file extensions that are to be watched.
port	`<int>`	`35729`	Port to be used to communicate file system changes.

Parameter	Type	Default value	Description
debug	<boolean>	false	Set this parameter to get more log information.

Proxy

The proxy middleware provides you with the capabilities to connect to different backend systems or to switch the SAPUI5 version of the application.

Configuration Examples

- **Connect to a backend system**

  To forward any request starting with the `path` parameter to the provided backend `url`, execute `npx fiori run` in your project with the configuration below in the `ui5.yaml` file.

```
- name: fiori-tools-proxy
  afterMiddleware: compression
  configuration:
    backend:
    - path: /sap
      url: https://my.backend.com:1234
```

- **Connect to a backend system with destination**

  If you use a destination to connect to your backend system, you can also provide the `destination` in the configuration.

```
- name: fiori-tools-proxy
  afterMiddleware: compression
  configuration:
    backend:
    - path: /sap
      url: https://my.backend.com:1234
      destination: my_backend
```

- **Connect to multiple backend systems**

  Additionally, you can connect to multiple backend systems as follows:

```
- name: fiori-tools-proxy
  afterMiddleware: compression
  configuration:
    backend:
    - path: /northwind
      url: https://my.backend_2.com:1234
    - path: /sap
      url: https://my.backend.com:1234
```

- **Connect to an ABAP Environment on SAP Business Technology Platform**

  If you want to connect to an ABAP Environment on SAP Business Technology Platform, you need to set the optional property `scp` to `true`. For any other target, remove this property or set it to `false`.

```
- name: fiori-tools-proxy
  afterMiddleware: compression
  configuration:
    backend:
    - path: /sap
      url: https://my.steampunk.com:1234
      scp: true
```

- **Connect to the SAP API Business Hub**

  If you want to connect to the SAP API Business Hub, you need to set the optional property `apiHub` to `true`, and set the corresponding `path` and `url`.

```
- name: fiori-tools-proxy
  afterMiddleware: compression
  configuration:
    backend:
    - path: /s4hanacloud
      url: https://api.sap.com
      apiHub: true
```

- **Proxy WebSockets**

  If you want the proxy to handle WebSockets, then you need to set the optional property `ws` to `true`.

```
- name: fiori-tools-proxy
  afterMiddleware: compression
  configuration:
    backend:
    - path: /sap
      url: https://my.backend.com:1234
      ws: true
```

- **Change the path to which a request is proxied**

  It is possible to configure the proxy to send requests from a certain path `/services/odata` to a destination with a specified entry path `/my/entry/path`. To do so, use the following configuration:

```
- name: fiori-tools-proxy
  afterMiddleware: compression
  configuration:
    backend:
    - path: /services/odata
      pathPrefix: /my/entry/path
      url: https://my.backend.com:1234
      destination: my_backend
```

- **SAPUI5**

  By using the proxy configuration, the user can also change the SAPUI5 version, which is used to preview the application. The initial SAPUI5 configuration for the proxy is created along with the application generation. See the following example:

```
- name: fiori-tools-proxy
  afterMiddleware: compression
  configuration:
    ui5:
      path:
      - /resources
      - /test-resources
      url: https://sapui5.hana.ondemand.com
      version: 1.78.0
```

  By using the `version` parameter, the user can select the SAPUI5 version which is used when `npx fiori run` is executed.

Serve Static

The serve static middleware provides the capability to serve any static resources locally from your machine. For example, you can serve SAPUI5 locally or any other resources.

Example Configuration for serving locally SAPUI5

Prerequisites

SAPUI5 SDK version is downloaded and extracted locally on your computer. You can download the SAPUI5 sources from the SAP Development Tools page.

If you want to serve the SAPUI5 resources from your computer, execute `npx fiori run` in your project with the configuration below in the `ui5.yaml` file. Any request starting with the `path` parameter is forwarded to the local path provided in the `src` parameter.

```
server:
  customMiddleware:
  - name: fiori-tools-servestatic
    afterMiddleware: compression
    configuration:
      paths:
        - path: /resources
          src: "Path/To/SAPUI5-SDK"
        - path: /test-resources
          src: "Path/To/SAPUI5-SDK"
```

Example Configuration for serving any resources locally

If you want to serve any resources from your computer, execute `npx fiori run` in your project with the configuration below in the `ui5.yaml` file. Any request starting with the `path` parameter is forwarded to the local path provided in the `src` parameter.

```
server:
  customMiddleware:
  - name: fiori-tools-servestatic
    afterMiddleware: compression
    configuration:
      paths:
        - path: /images
          src: "Path/To/images"
        - path: /libs
          src: "Path/To/libs"
```

Use Run Control

VS Code

Use *Run Control* (CTRL + Shift + D) in VS Code for different options of running your application.

- Start **project name**. Runs the application based on the SAPUI5 version selected during the generation.
- Start **project name** with SAPUI5 Version. Runs the application with the possibility to select from a list of SAPUI5 versions.

> **Note**
>
> After the SAPUI5 version is selected, the application will start automatically.

- Start **project name** Mock. Runs the application using mock data by executing the `npm run start-mock` script. Only available in OData V2 service.
- Start **project name** Mock with SAPUI5 Version. Runs the application using mock data and enables selection of an SAPUI5 version. Only available in OData V2 service.
- Start **project name** Local. Runs the application using mock data against a local copy of the SAPUI5 library that was selected during generation.

In VS Code, you can aslo create additional launch configurations. To know more, see Create a New Run Configuration in Visual Studio Code.

> **i Note**
> Using the run tool in VS Code is specific to the open workspace. If you change your workspace or folders open in VS Code, your run options may no longer be available.

SAP Business Application Studio

1. In SAP Business Application Studio, navigate to the Run Configuration pane:
   - One the left-side toolbar, click the **Run Configuration** icon.
   - On the main menu, click **View** > **Run Configuration**.
2. In the Run Configuration pane, see the list with different options of running your application:
   - Start **project name**. Runs the application based on the SAPUI5 version selected during the generation.
   - Start **project name** Mock. Runs the application using mock data by executing the `npm run start-mock` script. Only available in OData V2 service.
   - Start **project name** Local. Runs the application using mock data against a local copy of the SAPUI5 library that was selected during generation.

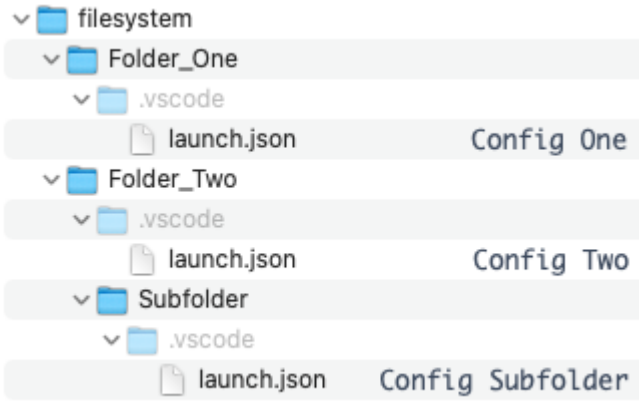
   In SAP Business Application Studio, you can also create additional run configurations. To know more, see Create a New Run Configuration in SAP Business Application Studio.
3. Select the required option of running your application and click the **Run Module** icon.

Run Control Overview

The **Run** dialog in VS Code as well as in SAP Business Application Studio looks for the file `<workspace_root>/.vscode/launch.json`, but won't traverse into any subfolders and consequently will not find any configurations in `launch.json` files, that reside in nested folders. It is possible to merge configurations from multiple`launch.json` files by using workspaces. Here is a sample that shows different development environment setups.

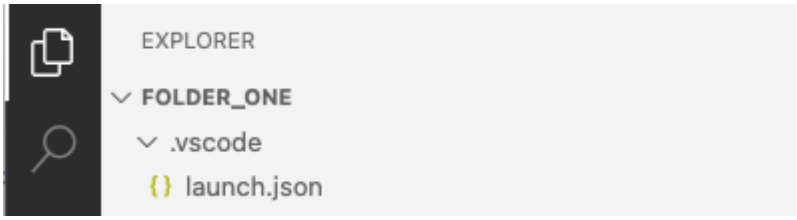
Example

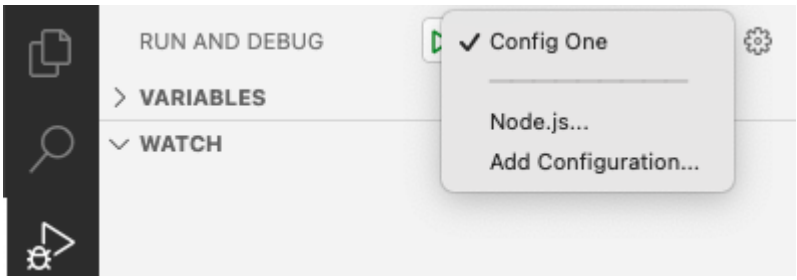
Assuming the following file system structure:



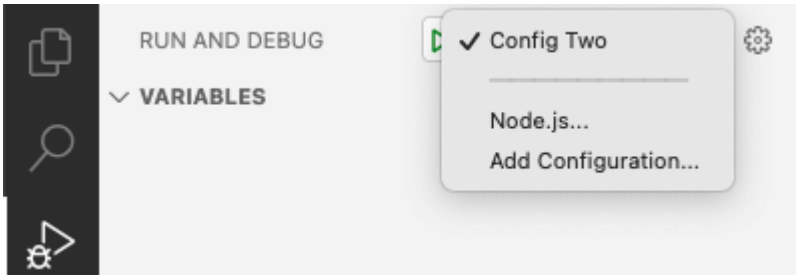
The `launch.json` files contain the configuration noted next to them: **Config One**, **Config Two**, and **Config Subfolder**. In VS Code you can open a folder in via `File->Open`. Regardless in which environment you open the folder, VS Code or SAP Business Application Studio, depending on which folder you choose, you will get different results.

When opening **Folder_One**, **Config One** is shown in *Run and Debug* view.





If you open **Folder Two** the *Run and Debug* view shows **Config Two**.



ⅈ Note

The config from Subfolder isn't shown in this case, as the configuration file `launch.json` can't be found in `<workspace_root>/.vscode`.

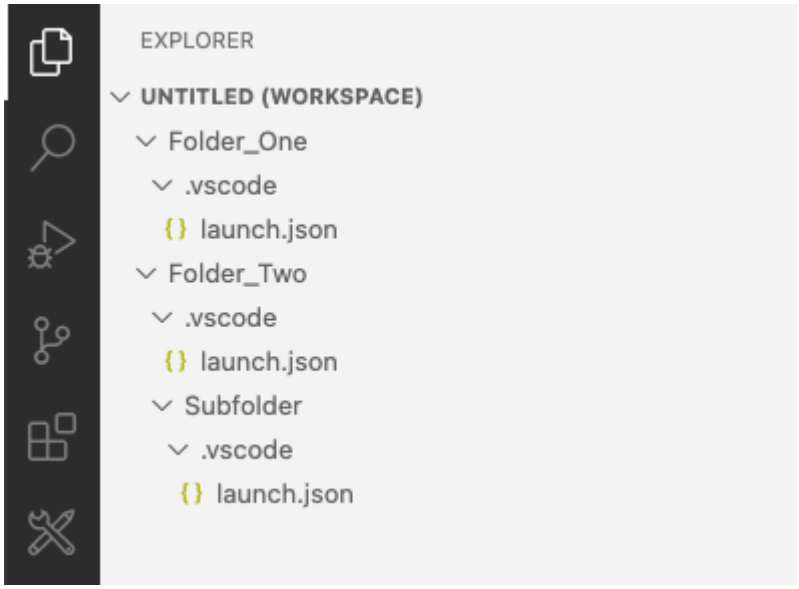
When you open **Subfolder** in your development environment, only **Config Subfolder** would be shown.

In addition, you can create a workspace, by opening one of the folders and selecting `Add Folder to Workspace` in your development environment. The workspace could be configured in many ways, here are some examples:

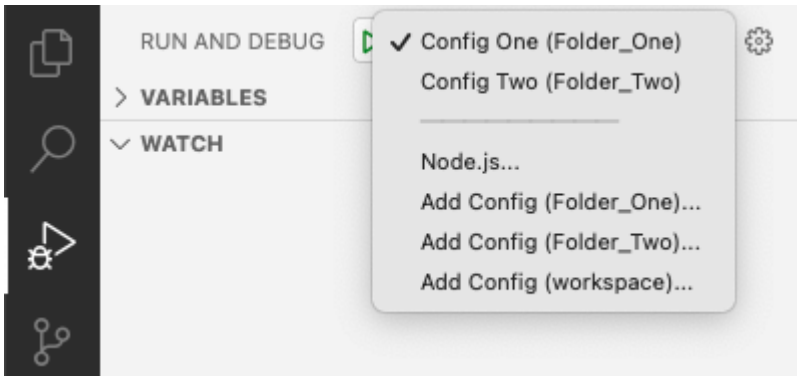
- **workspace root:** Folder_One

Only **Config One** is displayed, same as shown above in Open Folder_One.

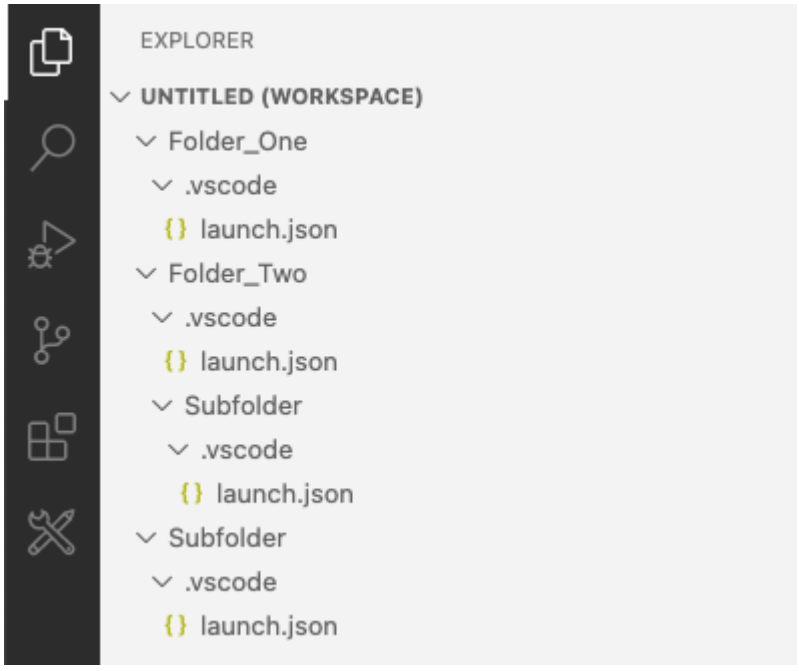
- **workspace roots**: Folder_One, Folder_Two



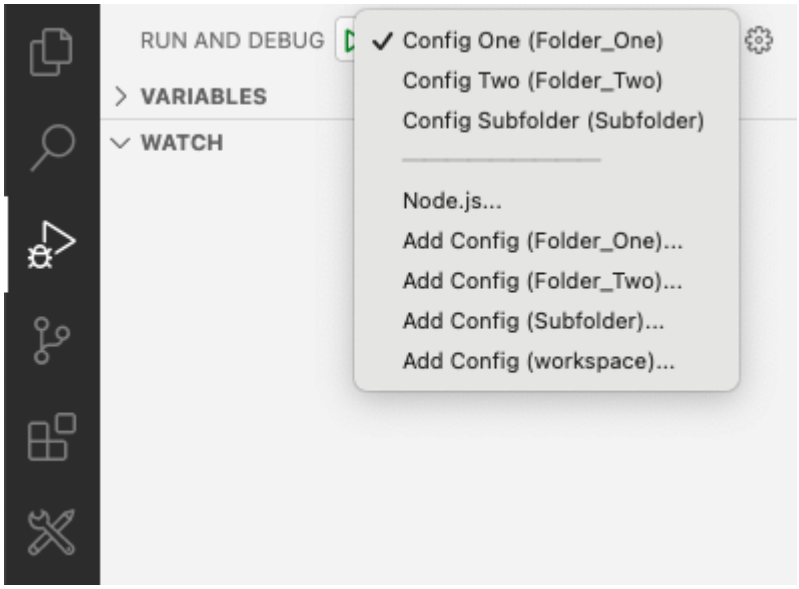
in this case Config One and Config Two are displayed, because these two configurations can be found in `Folder_One/.vscode/launch.json` and `Folder_Two/.vscode/launch.json`



- **workspace roots**: Folder_One, Folder_Two, Subfolder



in this case configurations from all three `launch.json` files are displayed:



ℹ Note

As seen in the last example, it's possible to add a subfolder of an existing workspace root folder as stand-alone workspace root.

For more information on run configurations, refer to.Launch configurations

Related Information

Create a New Run Configuration in Visual Studio Code
Create a New Run Configuration in SAP Business Application Studio

Create a New Run Configuration in Visual Studio Code

ℹ Note

Run Configuration Wizard requires @sap/ux-ui5-tooling version 1.5.3 and higher.

A new run configuration is created with the command *Fiori: Open Run Configurations*. To add custom configuration files for your project, perform the following steps:

1. From the command palette, select *Fiori: Open Run Configurations*.
2. Select the project that you want to add a new configuration and click Enter to confirm your input.
3. Existing run configurations for the selected project is displayed. Click Create.
4. Fill in the following sections:
    - **Name** (**mandatory**) - You can change the name.
    - **File Name** (**mandatory**) - Select html file that is used when application is started.
    - **Mock Data** - Runs your application with a mock server which will mock the OData requests.
    - **Support Assistant** - Enables application developers to check whether their applications are built according to the best practices for building SAPUI5 applications. The tool uses a set of pre-defined rules to check all aspects of an application.

- URL Components - Enables application developer to define additional URL parameters and/or hash fragment for the SAP Fiori launchpad intent-based navigation.
- Advanced Setting- Enables the application developer to define which SAPUI5 version will be used during runtime and/or change the destinations that are used by the application.

  Additionally the application developer can choose to download the SAPUI5 sources for a specific version by selecting Use local SAPUI5 sources. The downloaded SAPUI5 libraries are then used when running the preview. When Use local SAPUI5 sources is selected for preview, then the option Run with mock data is automatically selected.

Click Save

5. The new launch configuration appears in the Run and Debug pane on the left and also in the table on the UI.
6. To run the project, click on the > - icon in the Actions section of the table. Alternatively you can select Run Configuration from the drop-down list in the Run and Debug pane and click the Start Debugging icon.

Create a New Run Configuration in SAP Business Application Studio

ℹ Note

Run Configuration Wizard requires @sap/ux-ui5-tooling version 1.5.3 and higher.

In SAP Business Application Studio, you can create additional run configurations that define how your project is executed. To do so, perform the following steps:

1. From the left-side toolbar, click Run Configurations.

   The run configuration pane appears.
2. In the left pane, click + (the *Create Configuration* icon) to create a new configuration.

   A dialog box appears.
3. In the option What would you like to run?, you are prompted to select the project for which you want to create the configuration.
4. Fill in the following sections:
   - Name (**mandatory**) - You can change the name.
   - File Name (**mandatory**) - Select htm file that is used when application is started.
   - Mock Data - Runs your application with a mock server which will mock the OData requests.
   - Support Assistant - Enables application developers to check whether their applications are built according to the best practices for building SAPUI5 applications. The tool uses a set of pre-defined rules to check all aspects of an application.
   - URL Components - Enables application developer to define additional URL parameters and/or hash fragment for the SAP Fiori launchpad intent-based navigation.
   - Advanced Settings - Enables the application developer to define which SAPUI5 version will be used during runtime and/or change the destinations that are used by the application.

     Additionally the application developer can choose to download the SAPUI5 sources for a specific version by selecting Use local SAPUI5 sources. The downloaded SAPUI5 libraries are then used when running the preview.

When Use local SAPUI5 sources is selected for preview, then the option Run with mock data is automatically selected.

Click Save

5. The new launch configuration appears in the Run and Debug pane on the left and also in the table on the UI..
6. To run the project, click on the > - icon in the Actions section. Alternatively you can select the Run Configuration from the drop-down list in the Run and Debug pane and click the Start Debugging icon.

Additionally, in the Run Configuration Pane you can perform the following actions:

- **Bind/Unbind SAPUI5 Version:** Quick actions for changing the UI5 version of the run configuration.
- **Bind/Unbind Data Source:** Quick action for changing the destination of the run configuration.
- **Rename:** right-click on the run configuration and choose Rename to provide a new name for the selected run configuration.
- **Show in File:** right-click on the run configuration and choose Show File to open the JSON file containing the set of configuration properties, with the name highlighted.
- **Delete:** right-click on the run configuration and choose Delete to delete the run configuration.

App-to-App Navigation Preview

With the command `Fiori: Enable App-to-App Navigation Preview`, you can enable the preview function to follow a configured external navigation from one application to another if both applications are located in the same workspace.

To learn how to configure external navigation with SAP Fiori elements, follow this link.

1. In VS Code (CMD/CTRL + Shift + P) or SAP Business Application Studio, open **Command Palette** and enter Fiori: Enable App-to-App Navigation Preview.
2. Select the source application from where the navigation shall originate.
3. Select the target application to which the navigation shall lead.

   As a result, the following message is displayed: **App-to-App Navigation enabled**.
4. Start the preview of the source application and follow the configured external navigation.

The command generates a new configuration `appconfig\fioriSandboxConfig.json` to the source application folder and updates the `ui5.yaml` file. Also, it is possible to add multiple target navigations to the same source application.

Preview an Application on External Fiori Launchpad

This feature provides the user with the ability to test an application run without its redeployment. Running an application on the existing SAP Fiori launchpad requires the application to be deployed once and configured, so it's visible on the target launchpad .

ℹ Note

Currently, the SAP Launchpad service on SAP BTP doesn't support the use of this feature.

Usage

To run the application on the external SAP Fiori launchpad, perform the following steps:

1. Open the Command Palette Ctrl + Shift + P .
2. Use the **Fiori: Add FLP Embedded Configuration** command.
3. Enter the BSP of the deployed application.

   Note

   The BSP needs to be entered in lowercase.

4. Enter the YAML, which contains the back-end configuration. Usually, it's the `ui5.yaml` file.
5. Enter the relative link to the SAP Fiori launchpad, such as `sap/bc/ui5_ui5/ui2/ushell/shells/abap/Fiorilaunchpad.html`.
6. Before starting preview you'll need to build your application fist by executing `npm run build`.
7. To start the application on the existing launchpad, right-click the project folder or any subfolder and select **Preview Application**.
8. Select the newly added start-embedded option.

The initial load of the application on the external launchpad takes longer than with the local preview options. Once the application is loaded and the UI5 resources are cached, the application is loaded quickly.

When any change is done, the application isn't refreshed immediately. You need to rebuild the application by executing the `npm run build` to see the changes on the UI. Once the build is executed, the application running on the external launchpad is automatically refreshed.

Developer Variant Creation

With variant creation provided by SAP Fiori tools, developers can create variants for applications or individual tables, which can be distributed together with the application. The variants are stored as SAPUI5 flexibility changes in the project's `webapp/changes` folder and packaged with the application during the build step.

Note

Variants store view settings, such as filter settings or control parameters. On the UI, these variants are referred to as views.

The feature is delivered with the *@sap/ux-ui5-tooling* node module and its preview feature. Developer variant creation is supported from the SAPUI5 version 1.90 (OData V2 based applications) and 1.84 (OData V4 based applications). Currently, *@sap/ux-ui5-tooling* supports only ABAP service- based projects.

To create development variants for your Fiori project with SAP Fiori tools perform the following steps:

1. Execute the `start-variants-management` from the **Preview Application** context menu.

   In case this script is not present, configure the feature for the first time:

   - Open the Command Palette Ctrl + Shift + P .
   - Use the *Fiori: Add Configuration for Variants Creation* command.

2. After the script is executed, a new browser tab appears. It displays the preview of the application switched to the UI adaptation mode.
3. For creating developer variants, follow the steps described in Creating and Adapting Views.

   Note

   Visibility and role assignment are not supported for developer variants.

4. Click **Save & Exit** to save the changes.
5. For each new variant, one or multiple SAPUI5 change files are created in the `webapp/changes` folder of your project. You can open each file and replace static texts of your application with translatable text, such as `{i18n>textKey}` and maintain the text in the corresponding *i18n* file of your project.

ℹ Note

For more technical information see `@sap/ux-ui5-tooling`

Preview an Application with the SAP Horizon Theme

We now include the latest SAP theme, Horizon, for applications you create in SAPUI5 versions 1.93.3 and 1.96.0 or higher. The Horizon theme isn't released for productive use. We have made it available as experimental to gather your feedback as we continue to evolve the SAP Fiori design system.

Existing applications and extensions that use SAP Fiori elements or standard SAPUI5 controls will work with the Horizon theme without the need for technical adoption effort. This benefit is part of the value of using SAP Fiori elements.

If you build custom controls, you need to check if they maintain design consistency when rendered with the Horizon theme. To learn more, see SAP's UI Technologies supporting the new Horizon visual theme of SAP Fiori.

To ensure you have a version of SAPUI5 compatible with the new Horizon theme in your application, perform the following steps:

- For a new SAP Fiori application using the SAP Fiori application generator, ensure that you have chosen a minimum SAPUI5 version compatible with the SAP Horizon theme on the project attributes step. Also ensure that **Advanced Options** are set to **Yes** on the project attributes step. Then, you should be able to choose **SAP Horizon (experimental)** from the UI5 Theme dropdown.
- For an existing SAP Fiori application, you can update the SAPUI5 version used by the application by modifying the SAPUI5 version detailed in the `ui5.yaml` file.

  #### ℹ Note

  If no version of SAPUI5 is specified in your `ui5.yaml` file, no update is required. In this case, the latest version of SAPUI5 is used and compatible with the SAP Horizon theme.

Use the SAP Horizon Theme

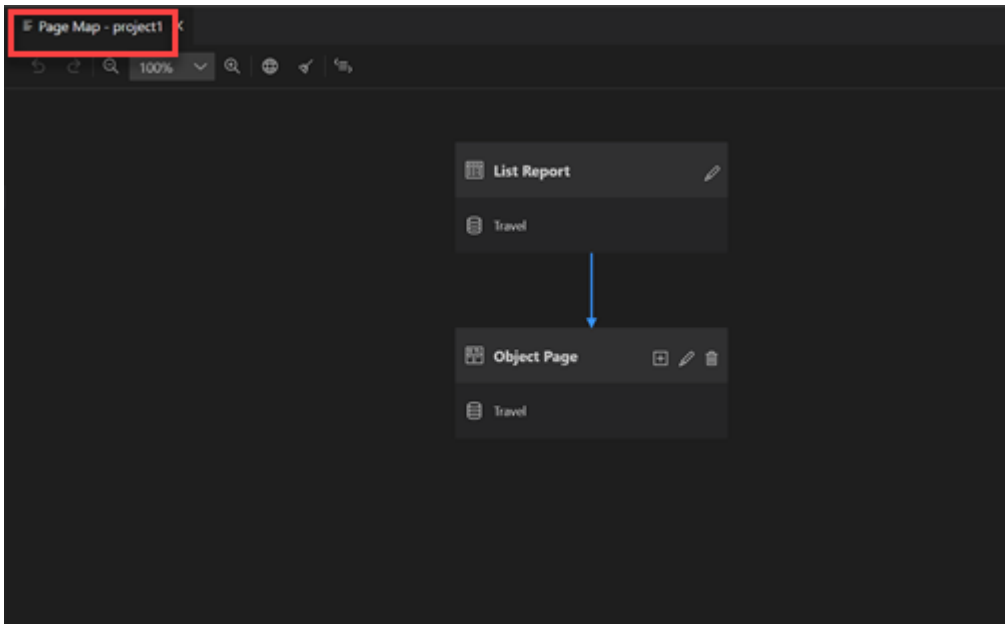
When you have a compatible version of SAPUI5 in your application, you can start using the new SAP Horizon theme. To learn more, see How to use the Horizon visual theme?

Develop an Application

Once the application has been generated, you can use SAP Fiori Tools – Application Modeler extension to preview and customize the SAP Fiori elements application.

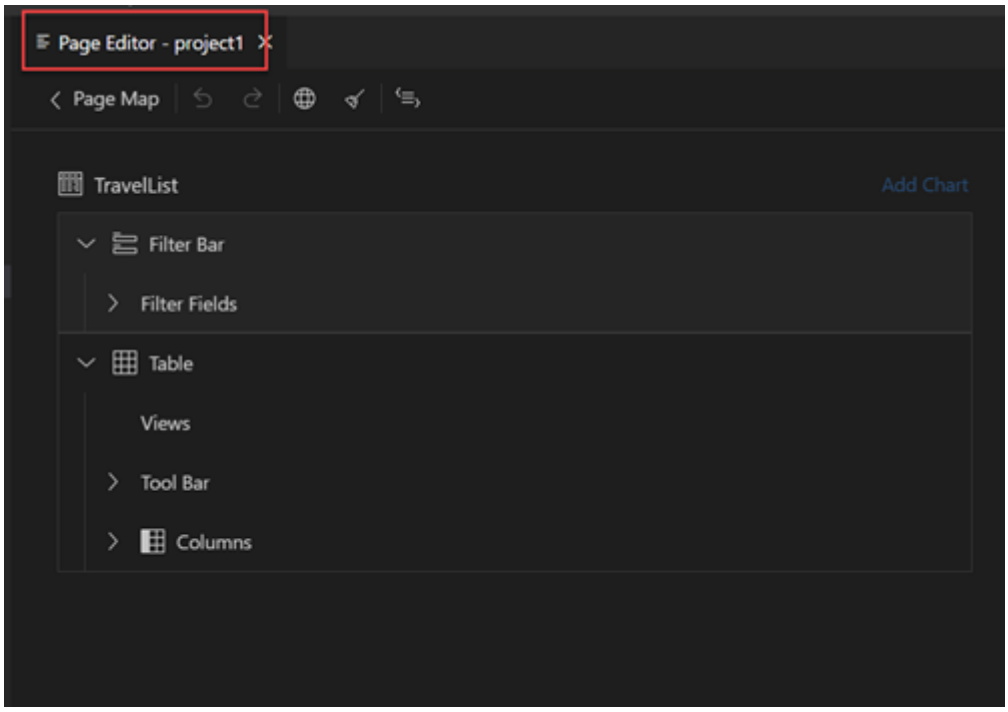
This extension provides the following capabilities.

- Define Application Structure. The **Page Map** provides a visual representation of the application pages, navigations, and the service entities that it uses. You can add new navigations and pages, delete pages, and navigate to corresponding editing tools. In addition, you can see global page settings that can be applied to the whole project.

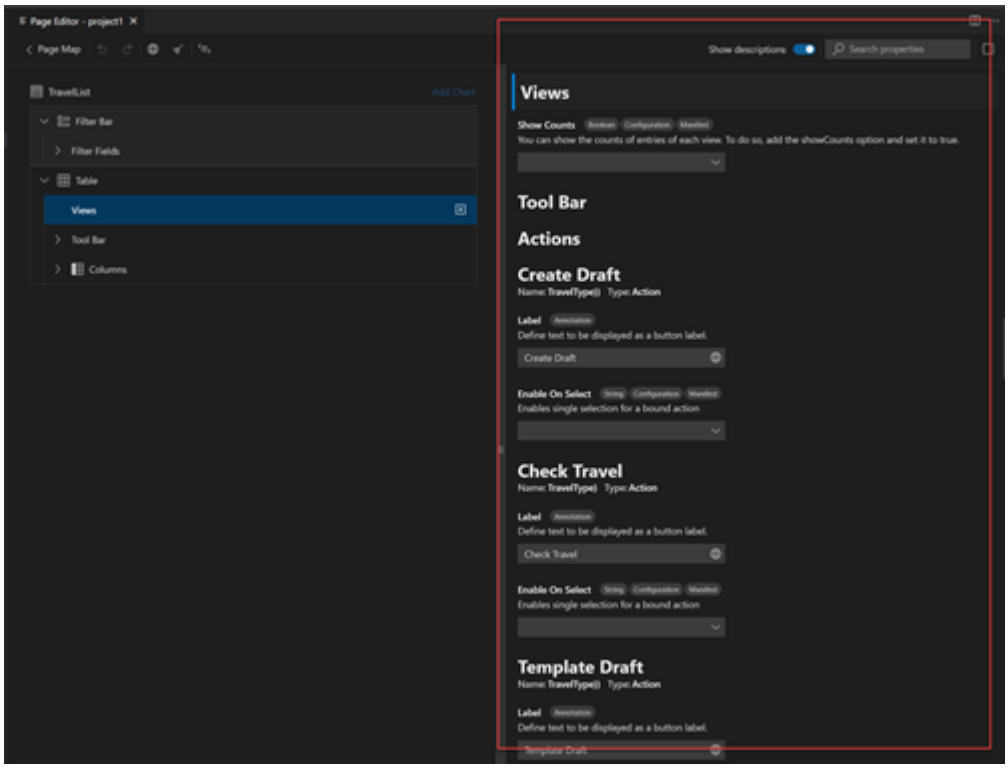


Click pencil icon to open **Page Editor**

- Configure Page Elements. The **Page Editor** provides an outline view of the configurable nodes on the selected page.



Once you select any node to change the settings, the **Property Panel** opens.



The saved changes are converted into corresponding artifact changes or UI flexibility changes in the project's folder of the application. Afterwards, the refresh of the preview is triggered.

Define Application Structure

Being part of the SAP Fiori Tools – Application Modeler extension, SAP Fiori tools **Page Map** allows to change the pages structure of the application and application-wide settings like the **Flexible Column Layout**.

Launching Page Map

You can launch **Page Map** in several ways:

1. From the Application Info page, see Application Information for more information.
2. **By using Command Palette**.
    - Open **Command Palette** and start typing ***Page Map***.
    - Select **Fiori: Show Page Map**.
    - In your workspace, select the SAP Fiori elements project.
3. **By using folder context menu**.

    If you already have a SAP Fiori elements project in your current workspace, right-click the project folder and select **Show Page Map**.

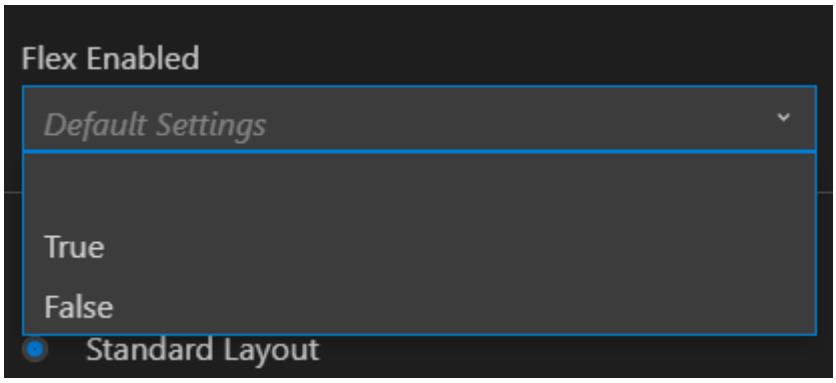
Using Page Map

Application Settings

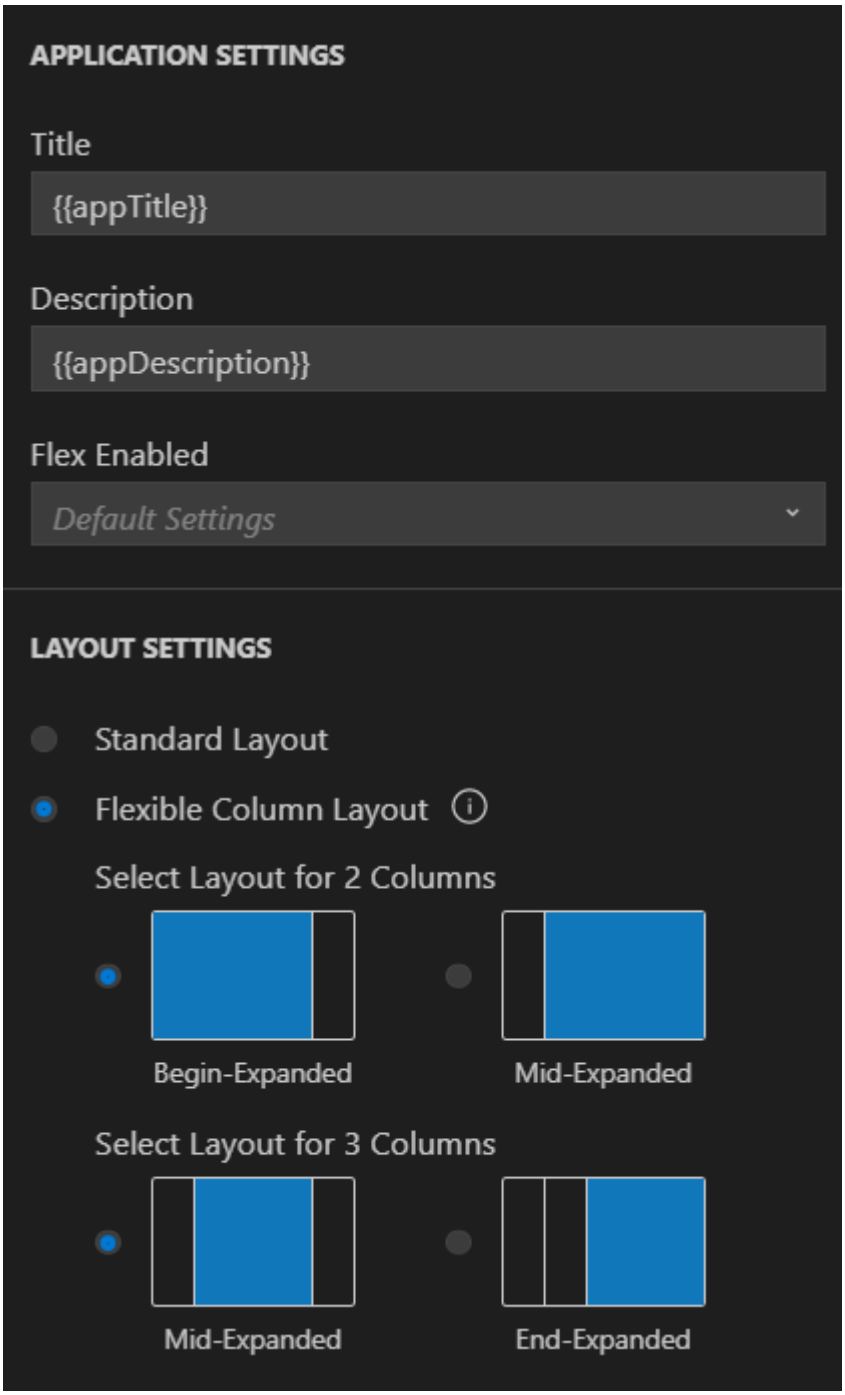
When the **Page Map** is open, you can see **Application Settings** in the default view. These settings contain the common and layout settings that are valid for the whole application.

- To change the application title after the application was created, use the **Title** field.
- To change the application description, use the **Description** field.
- The **Flex Enabled** field indicates whether the application supports UI adaptation. This field has the following options available:
    - *true*. Enables UI Adaptation.
    - *false*. Disables UI Adaptation.

  The default value is undefined.



  For more information, see SAPUI5 Flexibility: Enable Your App for UI Adaptation
- After you change any setting, the update applies immediately.
- To hide the panel, click **Toggle properties panel visibility** in the upper-right corner.



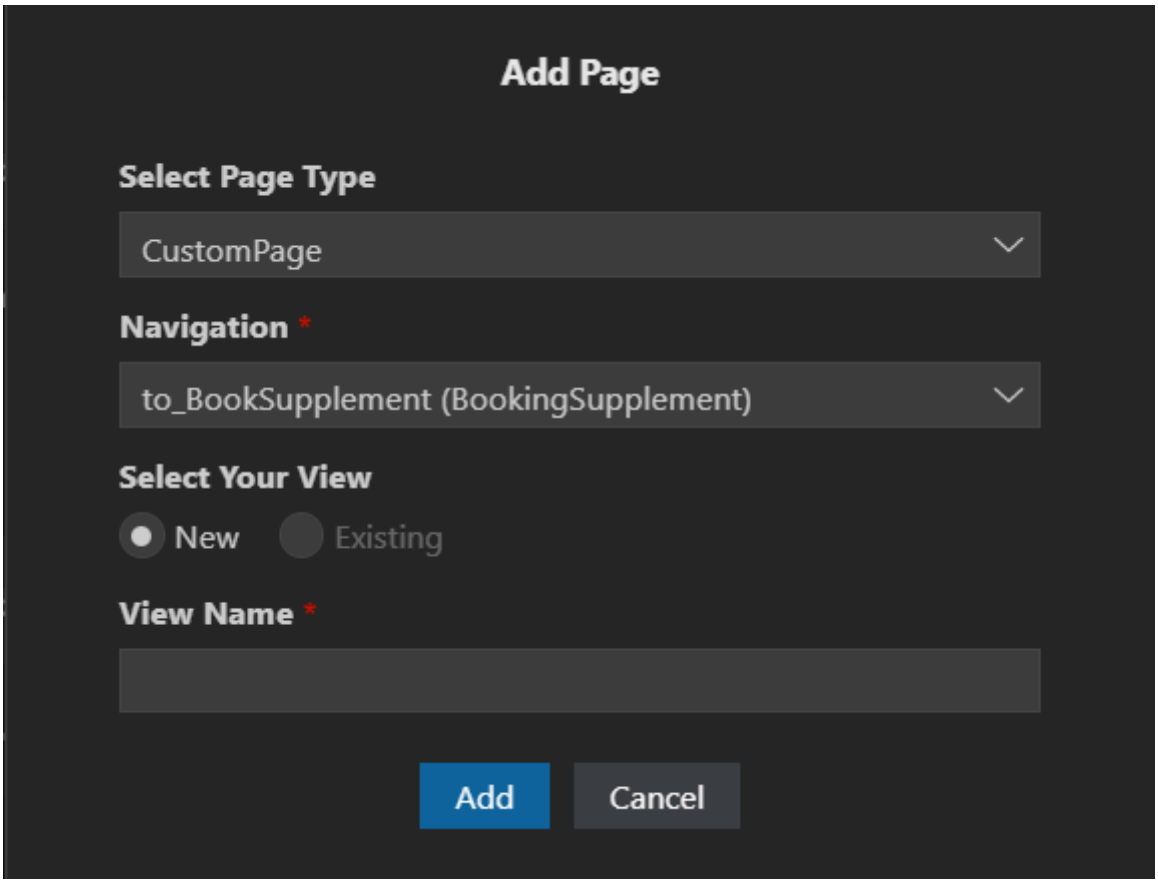
Adding a New Page

With the SAP Fiori tools **Page Map**, you can add additional pages to your application.

- Click the icon **Add New Page** in the header.
- From the **Select Page Type** list, select **ObjectPage** and click **Add**.
- In the **Navigation** field, select an entity that the page would navigate to.

  The list of available entities depends on the previous entity you navigate from. See Configure Page Elements for more information.

For `OData V4`, you can add a custom page.



- Click the icon **Add New Page** in the header of a page file.
- From the **Select Page Type** list, select **CustomPage** and click **Add**.
- In the **Navigation** field, select an entity that the custom page navigates to.
- Under **Select your view**, select one of the following option buttons:
    - **Create a New View**. Can create a new view.
    - **Use Existing View**. Provides a list of the prepared sample custom views.
- In the **View Name** field, enter the name of the view or select a value from the existing list.
- Click **Add**. As a result, a success message appears, such as **Custom Page ProcessFlow added successfully**.

ℹ Note

In case of `OData V4` based applications the templates used by the **Page Map** to provide this flexibility are published as part of our Open UX tools `@sap-ux/fe-fpm-writer` . This new transparency allows everyone to inspect the sources behind the scenes.

Configuring the Page

Click the **Configure Page** icon in the header of a page file to open the outline view of the Configure Page Elements. With the Configure Page Elements you can edit page properties . You can also open the Configure Page Elements from the tree view of the application modeler. When changes are applied, this is reflected in the `webapp/manifest.json` or the SAPUI5 flexibility changes are updated accordingly.

Deleting the Page

Click the **Delete Page** icon to delete the page.

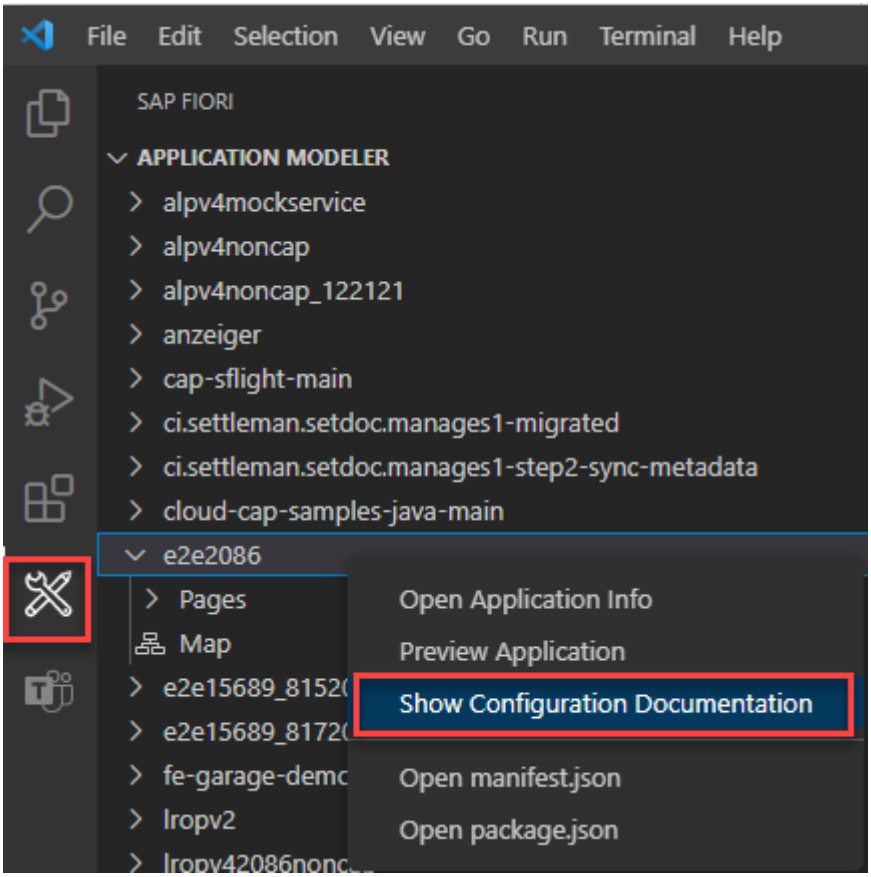
See Fiori Desing Guidelines: Layouts for more information about the Layout Settings.

Configure Page Elements

Developers can configure the SAP Fiori elements applications by using manifest settings and SAPUI5 flexibility changes (`OData` V2) in the **Page Editor**. The configuration properties available for UI-based maintenance are provided by the `@sap/ux-specification` node module, which is installed in the application's root folder.

There are different module versions that correspond to the different SAPUI5 versions. You can find the right version by checking the UI5-* tags at @sap/ux-specification . In case you need to change the **minUI5version** please see, Application Minimum SAPUI5 Version.

An overview of the available manifest and **UI5** flexibility properties can be accessed in the Application Information page. In addition, you can access the Configuration Documentation by right-clicking on your project in the tree view of the application modeler.



Page Editor Features

The **Page Editor** provides an outline view of the configurable elements on the selected page. To change settings, click on a node in the outline and the **Property Panel** will open. The **Property Panel** displays the editable properties, provides a search filter option, info tooltips for properties, and the option to edit the property directly in the associated file.

In the **Page Editor**, it's now possible to create and maintain annotation-based UI elements for **List Report**, **Object Page**, and **Form Entry Page** `OData V4` applications. With this feature available in the application modeler, application development becomes even easier and up-to-speed. For more information, see Maintaining Additional Elements.

Supported Templates of SAP Fiori elements

- **List Report Page** with `OData V2` and `OData V4`
- **Worklist Page** with `OData V2` and `OData V4`

- Analytical List Page with OData V2 and OData V4
- Overview Page with OData V2 and OData V4
- Form Entry Object Page with OData V4
- Custom Page with OData V4

How to Use Page Editor

To start using Page Editor, follow one of the available scenarios:

- Select the root folder of your app or any folder in your workspace within Explorer, right-click, and select Show Page Map. See Define Application Structure for more information about Page Map.
- Select the page that you want to configure and click the pencil icon.
- Navigate to the project sidebar view and click the respective page node in the tree view of the application modeler.
- In the text editor of the virtual JSON file of the page, click the Show Page Editor icon in the Editor Title menu.

Application Modeler Tree View

By default, the project is displayed with a simplified tree view. The tree view reflects the hierarchy of the virtual files, such as Project Name > App Name > Pages > Page Name.

To configure page layout and navigation, click the Map node that resides in the tree view. Then, the Define Application Structure opens at the right side of the tree view.

If you want to switch to the technical view displaying the full path with all the files generated in the background, you need to enable the JSON schema files. To do so, perform the following steps:

1. Navigate to Settings.
2. Select the Show JSON Schemas box.

As a result, the tree view of the application modeler is now updated with the generated JSON schemas files.

Adding Custom Column

ℹ Note

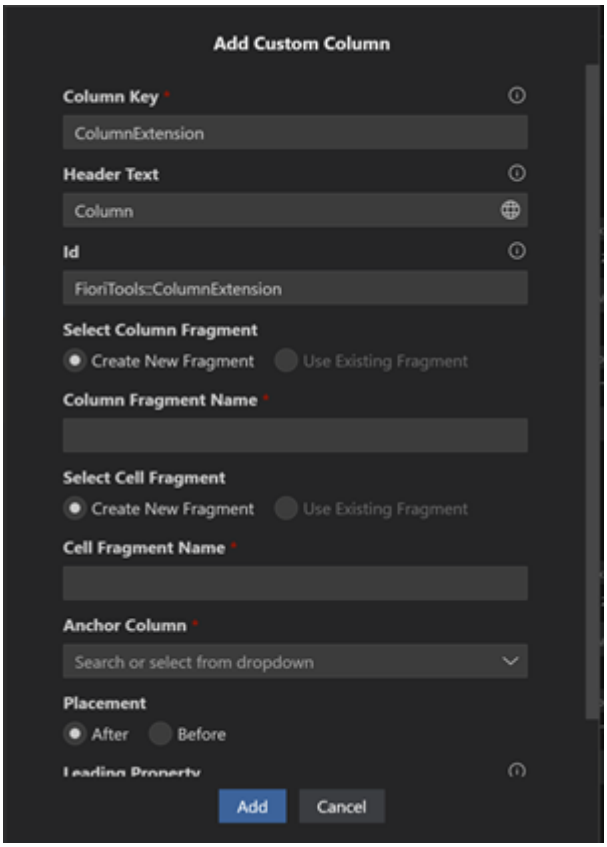
In case of OData V4 based applications the templates used by the Page Editor to provide this flexibility is published as part of our Open UX tools @sap-ux/fe-fpm-writer. This new transparency allows everyone to inspect the sources behind the scenes.

You have the ability to create a custom column for your table in a List Report or Analytical Chart page.

Custom Column (OData V2 only)

1. In the Page Editor outline view next to the Columns header, click the + icon to add a new column.
2. Provide the following information
    - **Column Key** - Key for the column
    - **Header Text** - column title
    - **ID** - unique ID is automatically created, but can be modified

- **Select Column Fragment**
  - Create New Fragment
  - Use Existing Fragment
- **Column Fragment Name**
- **Select Cell Fragment** (in case of responsive table type)
  - Create New Fragment
  - Use Existing Fragment
- **Cell Fragment Name**
- **Anchor Column** - select one of existing columns in the table. You can select where you want to insert the custom column, before or after the selected target column.
- **Placement** - Before/After
- **Leading Property** - If the content of your custom column refers to a property such as {Price}, you need to include a corresponding ***leadingProperty*** entry in the column definition.

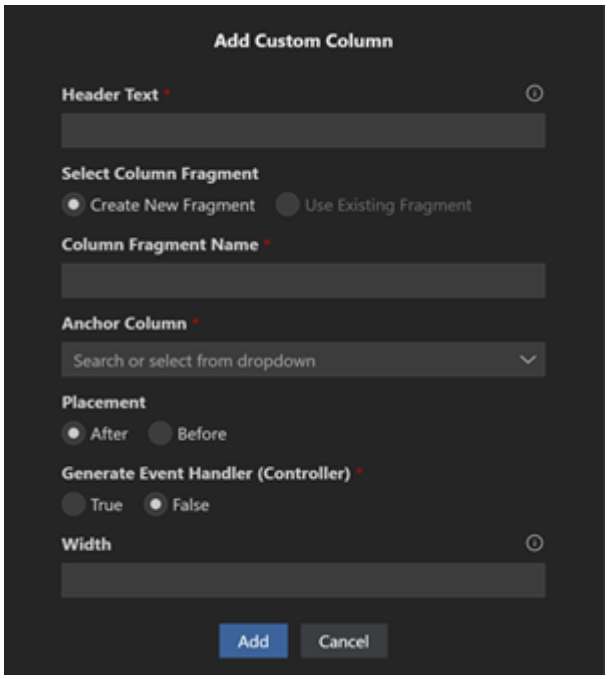


The custom column fragment and cell code is looked up and written to the project's `ext` folder. A custom column can be dragged into a new position using the handle in outline view. Click trash icon to delete a custom column.

Custom Column (OData V4 only)

1. In the Page Editor outline view next to the Columns header, click the + icon to add a new column.
2. Provide the following information
   - **Header Text** - column title
   - **Select Column Fragment**
     - Create New Fragment
     - Use Existing Fragment
   - **Column Fragment Name**

- **Anchor Column** - select one existing column in the table.
- **Placement** - Before/After. You can select where you want to insert the custom column, before or after the selected anchor column.
- **Generate Event Handler** - True/False
- **Width** - width of the new column



The custom column fragment and optional default controller code is written to the project's `ext` folder. A custom column can be dragged into a new position using the handle in outline view. Click trash icon to delete a custom column.

Adding Custom Section

ℹ Note

In case of `OData V4` based applications the templates used by the **Page Editor** to provide this flexibility is published as part of our Open UX tools `@sap-ux/fe-fpm-writer` . This new transparency allows everyone to inspect the sources behind the scenes.

You have the ability to create a custom section as part of your **Object Page** using the **Page Editor**.

1. In the **Page Editor** outline view of you're **Object Page**, click the + icon on the **Sections** node. For `OData V4` select Add Custom Section from the menu, for `OData V2` the selection will be pre-filled.
2. Provide the following information
   - **Title** - the label of the custom section
   - **View Type** - types are **View** or **Fragment** (`OData V2` only)
   - **Select Your Fragment/View**- either new or choose existing
   - **Fragment/View Name** - the file name of the artefact
   - **Anchor Section** - select one of the existing sections in the **Object Page**
   - **Placement** - before, after or replace (`OData V2` only)
   - **Generate Event Handler** - Decide whether a demo controller should be created (`OData V4` only)

On pressing Add the custom section fragment/view and/or controller code is written to the project's `ext` folder. A custom section can be dragged into a new position using the handle in outline view. Click trash icon to delete a custom section.

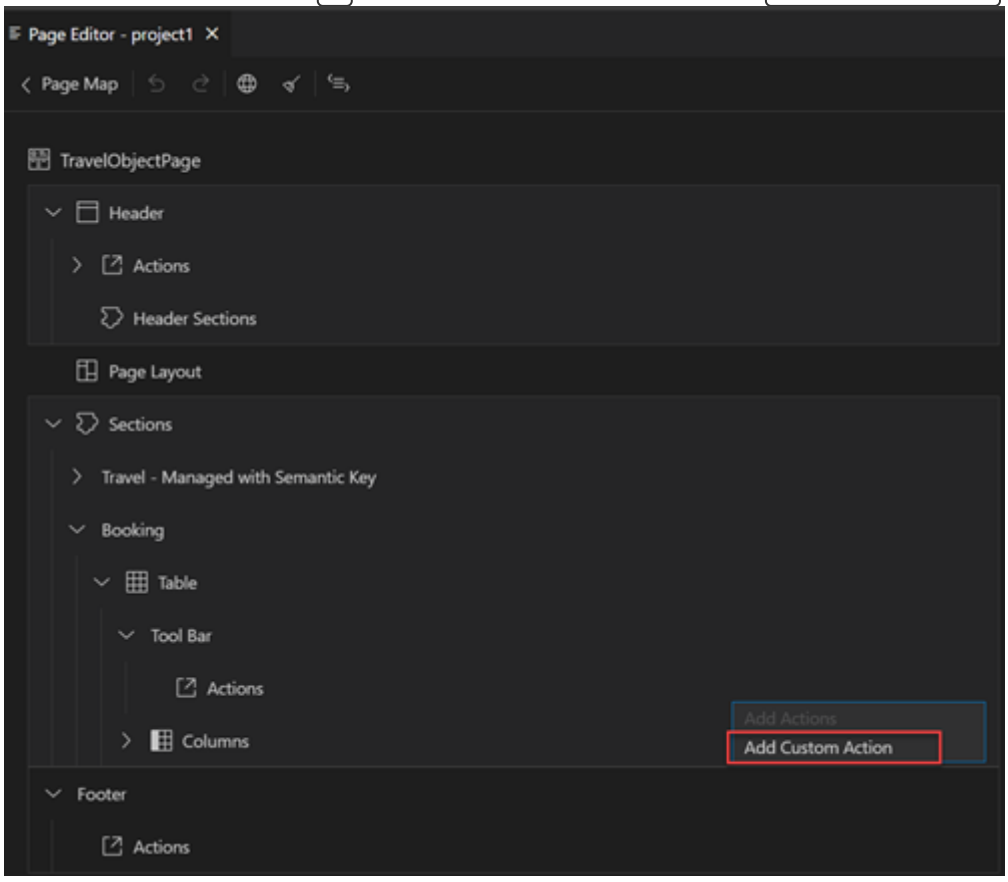
Adding Custom Action

ℹ Note

In case of `OData V4` based applications the templates used by the **Page Editor** to provide this flexibility is published as part of our Open UX tools `@sap-ux/fe-fpm-writer` . This new transparency allows everyone to inspect the sources behind the scenes.

You've the ability to create a custom action on in your **List Report** and **Object Page** using the **Page Editor** for `OData V4` applications.

1. In the **Page Editor**, click the + icon on the **Actions** node and select Add Custom Action from the menu.



2. Provide the following information:
    - **Action ID** - ID for the action
    - **Button Text** - text displayed on the button
    - **Anchor** - the key of another action to be used as placement anchor.
    - **Placement** - define placement after or before the anchor action.
    - **Action Handler File** - decide if you want to add to exiting file or create new action handler file.
    - **Handler File** - if select add to existing file, select the action handler file.
    - **Action Handler Method** - select if you want to create new function or add to existing function.
    - **Handler Method** - select handler method.

- **Required Selection** - toggle if this is required or not.

On pressing Add the custom action is written to the project's `ext` folder. A custom action can be dragged into a new position using the handle in outline view. Click trash icon to delete a custom action.

ℹ Note

This feature is only available for `OData V4` and with `@sap/ux-specification` version 1.96 or higher. See https://www.npmjs.com/package/@sap/ux-specification .

Adding Custom View

ℹ Note

In case of `OData V4` based applications the templates used by the **Page Editor** to provide this flexibility is published as part of our Open UX tools `@sap-ux/fe-fpm-writer` . This new transparency allows everyone to inspect the sources behind the scenes.

You can create a custom view in your **List Report** and **Object Page** using the **Page Editor** for `OData V4` applications.

1. In the **Page Editor**, click the + icon on the **View** node and select Add Custom View from the menu.

   ### ℹ Note

   The custom view feature is only available on **List Report** that doesn't contain a chart.

2. Provide the following information:

   - **Key** - Unique tab identifier.
   - **Label** - View title.
   - **Select Your Fragment** - Enter new fragment or choose an existing one.
   - **Fragment Name** - The file name of the artefact.
   - **Generate Event Handler** - Decide whether a demo controller should be created.

On pressing Add the custom view is written to the project's `ext` folder. A custom view can be dragged into a new position using the handle in outline view. Click trash icon to delete a custom view.

ℹ Note

This feature is only available for `OData V4` and with `@sap/ux-specification` version 1.96.29, 1.102.14 or higher. See https://www.npmjs.com/package/@sap/ux-specification .

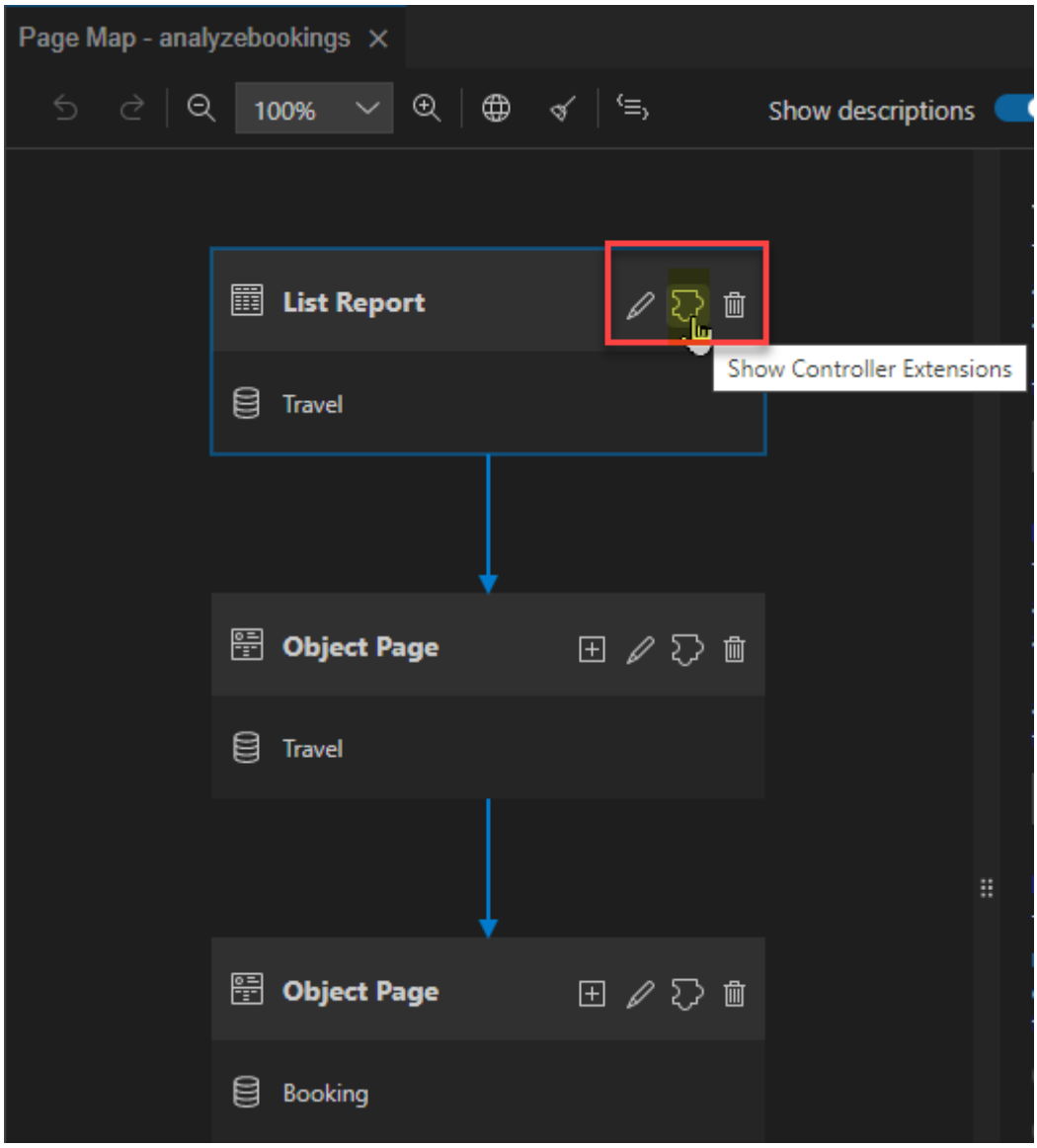
Adding Controller Extension

ℹ Note

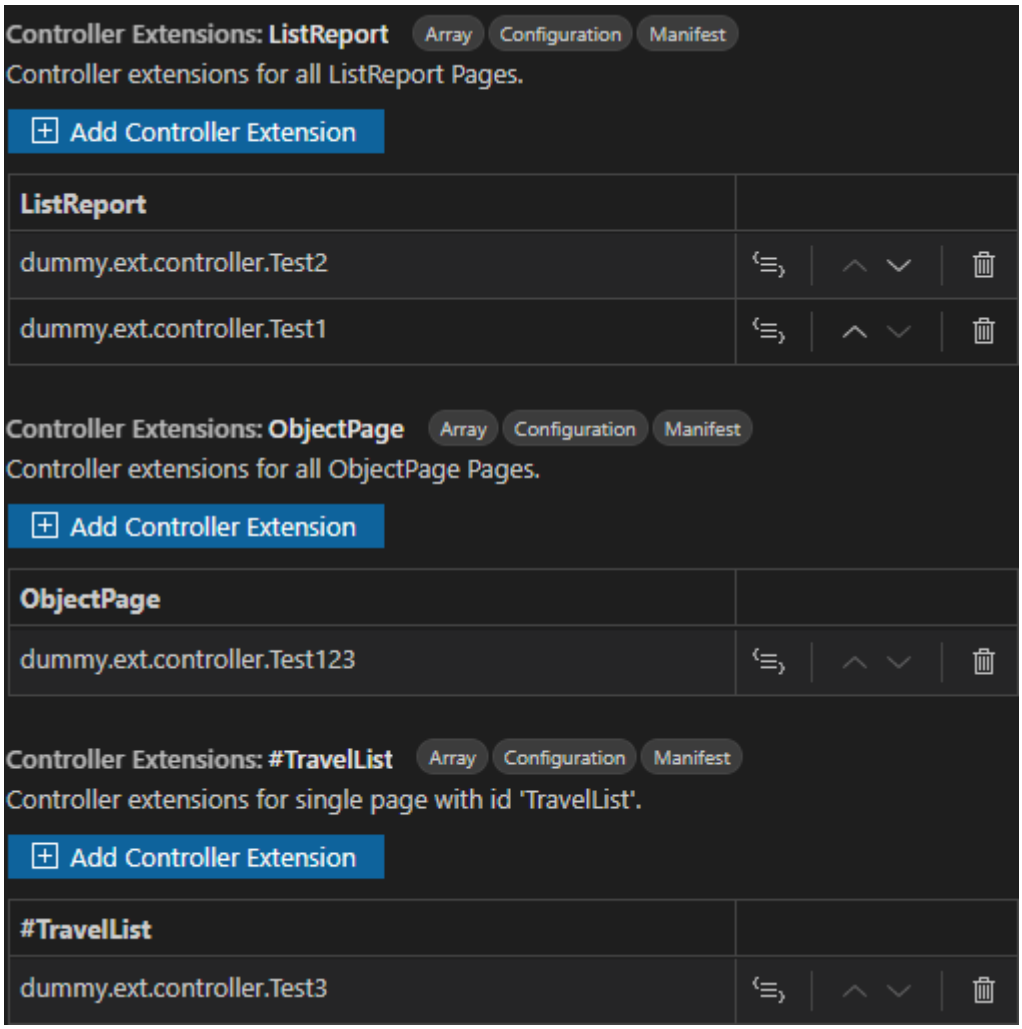
In case of `OData V4` based applications the templates used by the **Page Editor** to provide this flexibility is published as part of our Open UX tools `@sap-ux/fe-fpm-writer` . This new transparency allows everyone to inspect the sources behind the scenes.

You have the ability to create a controller extension as part of your **List Report** and **Object Page** using the **Page Map** for `OData V4` applications.

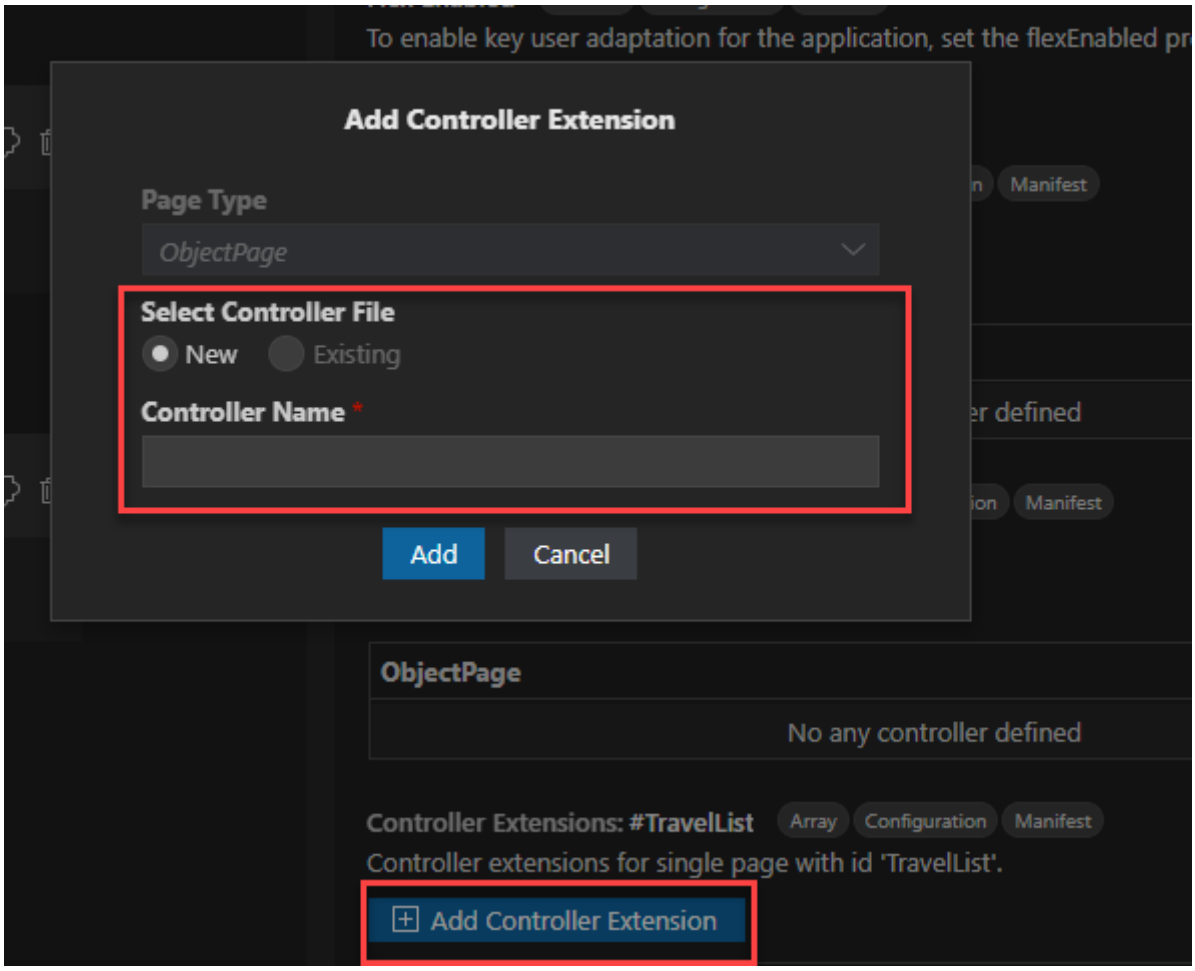
1. Launch the **Page Map**. For more information, see the section *Launching Page Map* in Define Application Structure.
2. In the **Page Map** view, click the Show Controller Extensions icon for your selected page.



3. You see the list of existing controller extensions for the selected page in the Properties Panel.



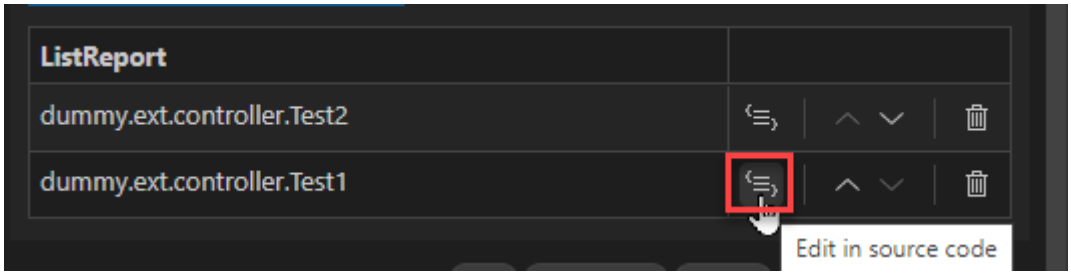
4. You can add a new controller extension by clicking + Add Controller Extension.



In the pop-up window, provide the required information.

5. You can then change the order in which the extensions are executed using the drag-and-drop functionality, or using the Move Up/Move Down icons.

6. You can also click **Edit in source code** to navigate to the respective controller code file.



→ Tip

For more information about controller extensions and related examples, see Controller Extensions.

Maintaining Additional Elements

ℹ Note

The annotation generation features within **Page Editor** is only supported for **List Report**, **Object Page** and **Form** applications based on the **OData V4** service. Contact SAP Support if any issue occurred. We rely on your feedback.

ℹ Note

The feature is provided since version 1.4.1 of the application modeler and @sap/ux-specification versions 1.84.25 and 1.90.14.

Application modeler provides the possibility to develop your application UI in a schematic view. In a Configure Page Elements, you see the basic structure outline of your page layout. You can add, remove, and modify properties of different page elements without knowing which annotations are used for that. The respective annotations are created and updated automatically in the local annotation file.

For example, you can add a section to the **Object Page** or **Form Entry Page** using the + button on the *Sections* node and a new `UI.FieldGroup` record is automatically generated and referenced in the `UI.Facets` annotation. If the latter isn't yet available in your application, it will be added. As a next step, you can add fields to the sections and define their properties, such as label, display type (including value help configuration), text, text arrangement, criticality, etc.

You can then use Edit in Source Code feature to see and manually edit the generated annotations. You can also always undo the changes made during the current session and redo them.

Overview

The following features apply to generated projects:

- Annotation Support
- Taskbar Notification
- Edit in Source Code

- Automatic Generation
- Internationalization (i18n)
- Project Cleanup

Annotation Support

The generated changes are always written to your application's single top-level local annotation file. Annotation files for modification are determined differently for CAP and Non-CAP projects.

CAP Project

Page Editor modifies the top level `.cds` file in the application folder. This file is identified as follows:

- **Page Editor** searches for a file in the application directory that isn't registered in `index.cds` or `service.cds`.
- If such a file doesn't exist, a new file `annotations.cds` is created with a `using` directive pointing to the service.

> **Note**
> If `index.cds` file doesn't exist, it's created and updated with the `using` directive pointing to the newly created annotation file (`annotations.cds`).

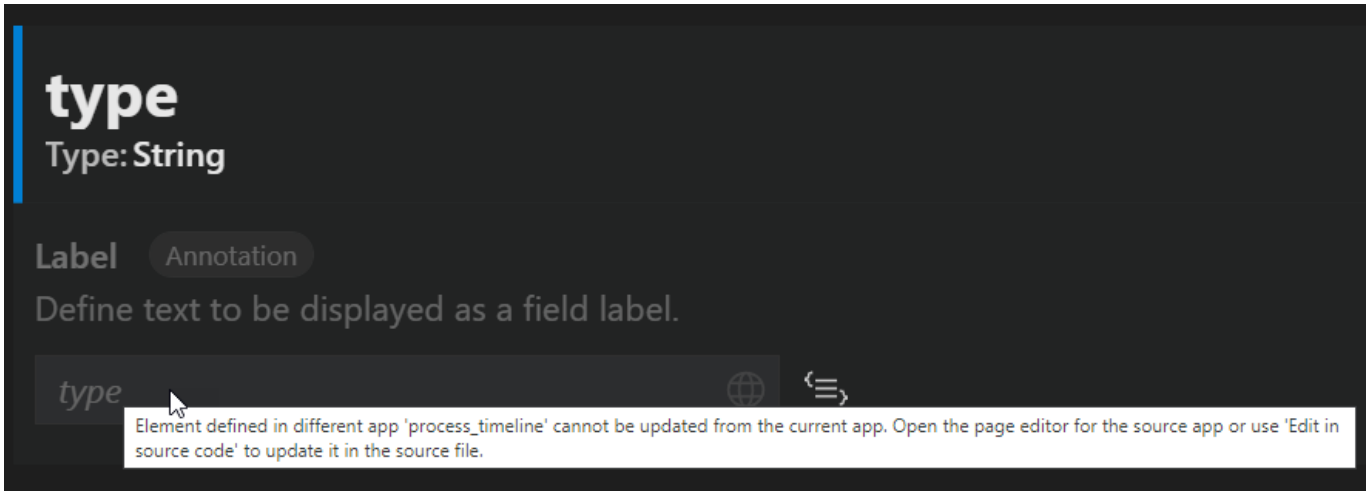
- If multiple files are found, top-level file is determined based on the `using` directives.
- If there are multiple files on the same top level, first found file on that level is being used.

Special handling is applied if the annotation to be modified resides in the different file:

- If it's defined in the **base** layer, that is, in the file set lower in the hierarchy, it's overridden in the top-level file.

> **Note**
> Some of the property values, such as measures and currencies, can't be overridden if defined in the base layer to keep the consistency across the project.

- If it's defined in the different `.cds` file of the same app that is on the same hierarchy level, it's being overridden in the file maintained by the **Page Editor** and a `using` directive is added to the overridden file to establish the layering hierarchy.
- If it's defined in the `.cds` file of the different app, it can't be overridden from the current app to avoid the cross references between apps. In this case, open the **Page Editor** for the original app to modify it. Use the tooltip to check the name of the original app:



Non-CAP Project

Page Editor modifies the local annotation file at the top of the annotation source hierarchy. The annotations in the lower level, such as service metadata or local annotation files at the bottom of the hierarchy are never modified by the Page Editor. If the changes made require modification to the annotation already existing in the lower layer, this annotation is copied to the top most local annotation file and edited there, thus overriding the annotation in the lower layer.

- The top of annotation source hierarchy is determined by the last entry in the `manifest.json` *<datasources => annotations file>*.

ℹ Note

Annotation file hierarchy can be viewed in the **Annotation File manager** and can be changed. Use `Fiori: Open Annotation File Manager` to see the **Annotation File manager**.

Example

Last entry in *<datasources => annotations file>* of `manifest.json` has the highest precedence.

- If the local annotation file doesn't exist, it's automatically created and registered in the `manifest.json` file as soon as the first annotation change is made.
- If a new file is created, it's placed under the `webapp` folder under the `annotations` directory.

Taskbar Notification

Taskbar notification is provided when the annotation file is updated based on the user actions in the Page Editor.

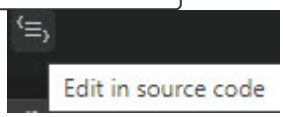
ℹ Note

The taskbar notification isn't displayed when the change is directly made in the annotation file.

Edit in Source Code

With the **edit in source code** feature, the user can navigate to the code fragments in the annotation file where the related annotation is defined.

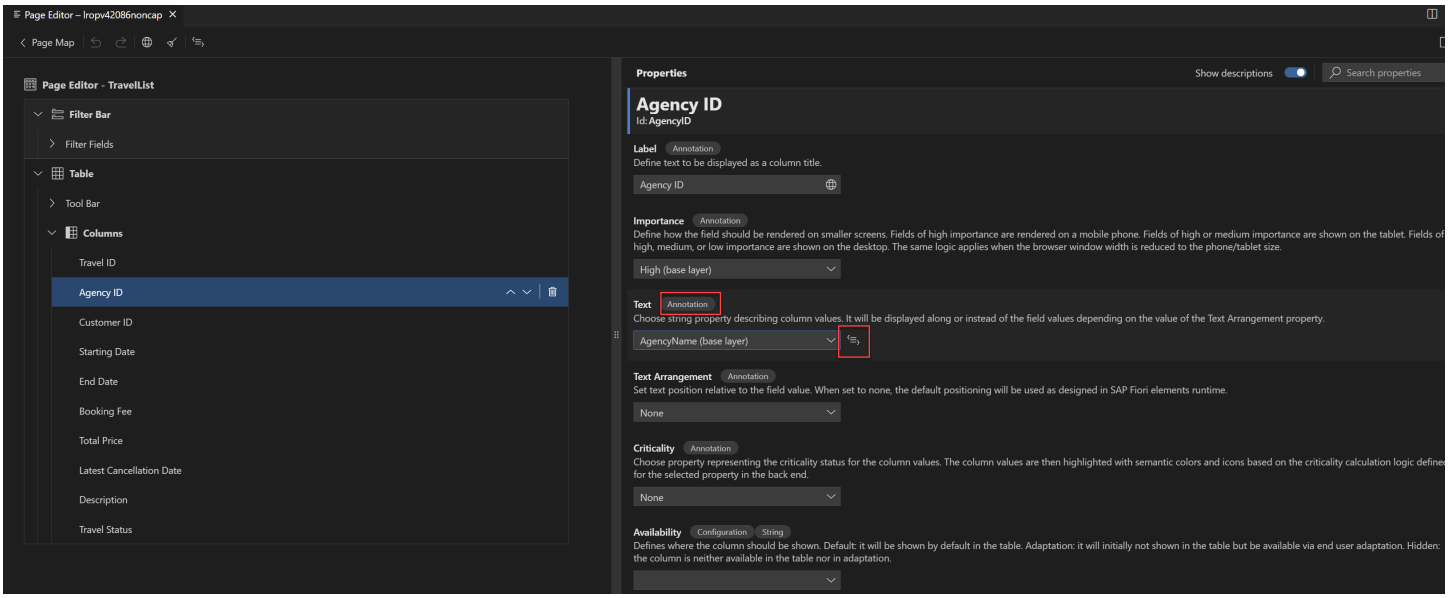
How to Use the Feature

To find the Edit in source code button in the property panel for editable and noneditable properties, locate the annotation tag, and click on the 

button next to the text field. If multiple definitions are present, a pop-up window with the options appears, and you can click a particular file to see the definition.

ℹ Note

Multiple definitions pop-up window is only available in VS Code. Only first annotation is shown in SAP Business Application Studio.

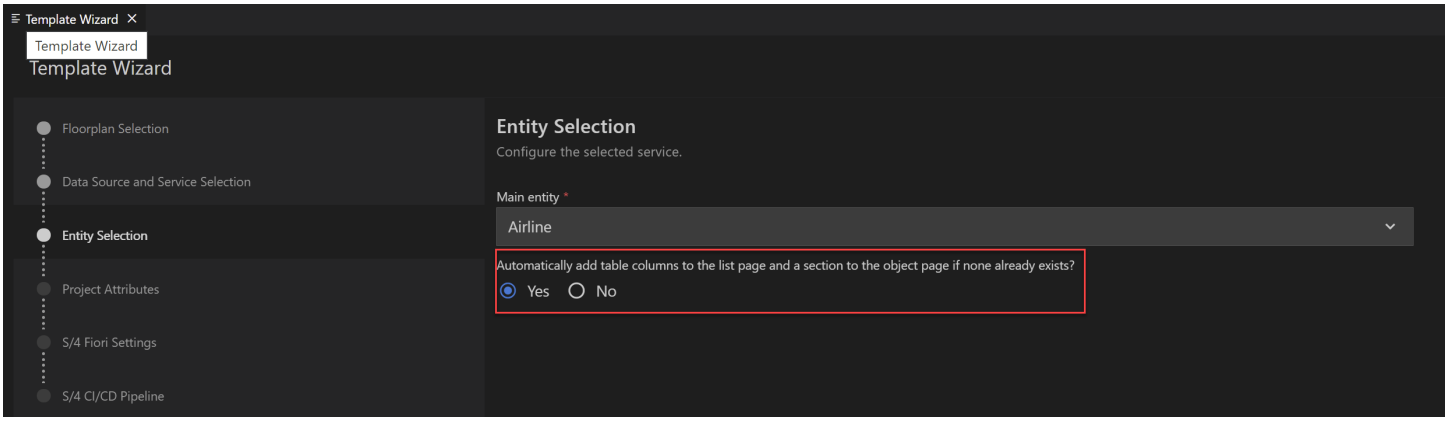


ℹ Note

There can be more than one way to configure a UI feature on the Fiori elements-based application. Some ways will be too complex for **Page Editor** to interpret. In those cases, the respective field control (dropdown, input field), including `Edit in source code` button isn't rendered in the properties panel. Instead, the link **edit in source code** is displayed with the same functionality as a button. You can then use this link to edit the property directly in the code.

Automatic Generation

When generating your application, you have the option of automatically adding page elements, such as **List Report** table or **Object Page** section. You can select the option `Yes` during **entity selection** step when generating your application. A default annotation is generated depending on the template and based on the following logic:



`UI.FieldGroup`, `UI.Facets`, and `UI.LineItem` annotation terms are generated for the Main Entity type and saved in the local annotation file. This only happens if these annotation terms aren't yet defined to avoid overriding.

- `UI.LineItem` annotation is generated for the list report and contains `UI.DataField` records referencing the first properties of the main entity type. Hidden properties and properties of type UUID are excluded.
- `UI.FieldGroup` and `UI.Facets` annotation referencing it in `UI.ReferenceFacet` records are generated on the main entity type for the **Form** and **Object Page**.
- `UI.FieldGroup` contains `UI.DataField` records for all the direct properties of the main entity type. Hidden properties are excluded.

Internationalization (i18n)

Internationalization, often abbreviated as i18n, is the process by which developers prepare a software product to be adapted in other languages. This topic covers the following aspects of managing i18n translation files for your projects:

- i18n Configuration
- Label Translation Cases
- Mass i18n Creation

To enable translation, click Internationalization (i18n) icon. 

When a new confirmation pop-up window appears, click Apply to confirm the action. Changes are applied to 18n property.

i18n Configuration

A resource bundle folder is also known as i18n folder and file names are maintained in the configuration as follows:

- Non-CAP projects at `/webapp/manifest.json`.
- CAP projects at `.cdsrc.json or package.json`. CAP has a configuration setting where the folder names can be used. A default value is ['_i18n', 'i18n', 'assets/i18n'].

Internationalizing UI Texts

Most UI texts in the application page, such as sections or field labels, are translation relevant and are eligible for internationalization. When you add elements that aren't directly based on an entity property, such as a section, you're asked to enter a desired text for the label directly in the **Add** dialog. When you add an element that is based on a property, such as section field or basic table column, you aren't prompted to enter a label text. Instead a label defined on the property is used if it's already defined or generated based on a property name. You have then the option to change this label for the given context in the **Property Panel**.

Independent on the origin of the translation relevant texts, whenever it appears in the **Page Editor** UI, e.g. properties pane or Add dialog, the Internationalization (i18n) icon appears on the right side of the input field. With this icon, you can handle the internationalization of the text string used in that specific field. It works as follows:

1. When the label isn't internationalized.

   In this case, clicking the Internationalization (i18n) icon, the following message appears with the user actions Apply and Cancel:

   ```
   Generate a text key <uniquekey> in the i18n file and substitute <actual text> by {i18n>uniqu
   ```

   With the Apply action, a new i18n key with text is generated and written to the i18n file, as well as referenced as a label.

   > **Note**
   >
   > If the **Field** or **Column** label is defined directly on the property with `@title` or `@Common.Label` annotations in the lower layer, the Label property is generated in the respective `UI.DataField` record with the reference to the i18n text in the resource bundle. The existing `@title` or `@Common.Label` annotations aren't modified or overridden.

   > **Note**

If the `UI.FieldGroup` or `UI.Facet` annotation is located in the lower layer, then it first creates a copy in the application layer.

2. When the label is defined as a reference to the text key but the text key is missing in the i18n files.

   In this case, when the user clicks the Internationalization (i18n) icon, the following message appears with the user actions Apply and Cancel:

   ```
   Generate a text key <uniqueKey> with value <uniquekey> in i18n file.
   ```

   With the Apply action, the i18n key, and text with the same value as the key is written to the i18n file.

3. When the label is defined as plain text but the text key for that text already exists in the i18n file.

   In this case, when the user clicks the Internationalization (i18n) icon, the following message appears with the user actions Apply and Cancel:

   ```
   Text key <uniqueKey> for value <sample text> is available in i18n file. Substitute <sample t
   ```

   With the Apply action, an existing i18n key is referenced as a label.

ⓘ Note

The i18n key is generated in camelCase for Non-CAP projects and PascalCase for CAP projects.

ⓘ Note

As you often need to change the labels several times during the application development and Internationalization (i18n) icon creates the new entry in the i18n file for each new text, you can end up with a number of unused i18n texts that will be provided for translation. To avoid the redundant keys and reduce the translation costs, it's recommended to handle i18n at the end of the development phase when labels are not likely to change.

Mass i18n Creation

In addition to the internationalization on the single text level, the icon for mass i18n generation is provided on the top of the screen. It reduces the effort for preparing multiple UI texts for translation.

Project Cleanup

The project cleanup procedure is defined with the icon **Cleanup**  and removes the following elements:

- All orphaned `UI.FieldGroup` and `UI.LineItem` with annotations, which aren't referenced as the `UI.ReferenceFacet` targets.
- Annotations with the terms `UI.MultiLineText`, `Common.ValueListWithFixedValues`, `Common.Text`, `Common.ValueList`, and `Common.FieldControl` are applied to entity properties that aren't mentioned in any referenced annotations.

Supported Elements in Page Editor

The following annotation-based UI elements are supported in the Configure Page Elements. All the other elements can be modified directly in the annotation file.

List Report Elements

The **List Report** lets the user work with a large list of items. It combines powerful functions for filtering and displaying results. The **List Report** consists of many elements, see List Report Elements.

ℹ Note

As stated above only these elements are supported by the Configure Page Elements, all other elements can be modified directly in the annotation file.

- Filter Fields
- Table

Object and Form Entry Page

The **Object Page** lets you display, edit, and create objects, as well as save drafts. It's suitable for both simple objects and more complex, multifaceted objects. The **Object Page** view gives you optimal support for multiple devices. The **Object Page** consists of many elements, see Object Page .

ℹ Note

As stated above only these elements are supported by the Configure Page Elements, all other elements can be modified directly in the annotation file.

- Header
- Form Section
- Table Section
- Identification Section
- Group Section
- Adding Custom Section
- Footer

Related Information

Maintaining Annotations with Language Server

List Report Page

The following links give you general information about the **List Report** page and what is supported by the Low Code Business Application :

- Fiori Design Guidelines
- Developing Low-Code Business Applications

The current supported elements of the **List Report**:

- Filter Fields

- Report Table
- Multiple Views
- Analytical Chart

Filter Fields

Filter Fields

Filters are located in the filter bar and can be represented by regular filters also known as compact filters and visual filters. Compact filters are represented in runtime as filter fields with value help, whereas visual filters are represented as charts with selectable elements. Visual filters are only available if your service is enabled for analytics.

ℹ Note

If you work with CAP (Node.js) project, note that some of the analytical features depend on the used OData parser. For more information, see Release notes CAP.

Adding Filter Fields

In the filter area, the user can add or remove **Filter Fields**. The **Filter Fields** are used to filter table entries in list report.

To add a **Filter Fields**, perform the following steps:

1. Navigate to **Page Map** > **List Report** to open a list report project.
2. Click **Configure page** (the pencil icon).

   Then, the **Filter Bar** appears.
3. Navigate your pointer over **Filter Bar** > **Filter Fields** or **Filter Bar** > **Compact Filters**
4. Click + to add a filter field.

   ### ℹ Note

   If your service is enabled for analytics but there are no visual filters in your **List Report**, choose **Add Compact Filters**.

   A new **Add Filter Fields** pop-up window with a list of available filter fields appears.
5. Search for or select the properties to be used as filters.
6. Choose **Add**.

While adding new **Filter Fields** the following logic applies:

- `UI.SelectionFields` annotation is generated or updated in local annotation file.

Adding Visual Filters

If your service is enabled for analytics, you can define visual filters represented as bar charts in runtime. To add a **Visual Filters**, perform the following steps:

1. Navigate to **Page Map** > **List Report** to open a list report project.

2. Click **Configure page** (the pencil icon).
3. Navigate your pointer **Filter Bar** > **Filter Fields** or **Filter Bar** > **Visual Filters**.
4. Click + to add a filter field.

> **Note**
>
> If your service is enabled for analytics but there are no visual filters in your **List Report**, choose **Add Visual Filters**.

A new **Add Visual Filter** pop-up window appears.

5. Search for or select the properties to be used as filters.

> **Note**
>
> If you select a time-based filter field, that is, a service property of type Edm.Date, Edm.Time, Edm.DateTime, or Edm.DateTimeOffset, you can choose the chart type bar or line. You can only use a bar chart to represent your visual filter for all other types.

6. Choose the entity that contains the appropriate filter values. Only analytically enabled entities are available for selection.
7. Choose the property representing the filter values. It's used as a dimension in the chart representing the visual filter. Only groupable properties are available for this selection.
8. Choose the measure for the chart representing the visual filter. You can use an existing measure, if available or create a new one.

> **Note**
>
> A new measure can be created based on the aggregated property of the selected value source and supported aggregation method as long as there's no existing measure based on the same combination. Creating new measures based on properties with custom aggregations aren't supported.

   - If you choose to use existing measure, select one of the available measures defined with custom or transformation aggregations in the Name field.
   - If you choose to create new measure, choose the aggregatable property and one of the supported aggregation methods.

9. Select **Add**.

A new visual filter is added with the basic properties. You can maintain additional properties of your visual filters in the **Property Panel**.

> **Note**
>
> Adding a visual filter updates the local annotation file and app descriptor file of your application.

The following annotations are generated in the local annotation file:

- `UI.Chart` based on the selected measure and dimension.
- `UI.PresentationVariant` referencing this chart.
- `Common.ValueList` annotation referencing the presentation variant via its qualifier.

The `manifest.json` is updated with the control configuration for `com.sap.vocabularies.UI.v1.SelectionFields` referencing the selected filter field along with the generated `Common.ValueList` annotation.

Moving Filter Fields

To move a field within the list of the, **Filter Fields** or **Compact Filters**, you can perform one of the following options:

- Drag and drop **Filter Fields** to the desired location.
- Press ↑ ↓ buttons in UI with a mouse.
- Use keyboard to set the focus on the ↑ ↓ buttons in UI and hit Enter,

The sequence of the property paths in `UI.SelectionFields` is adjusted which changes the sequence of the filters in the application preview.

You can't change the sequence of the visual filters directly, as it depends on the sequence of the compact filters defined in `UI.SelectionFields`. To change the sequence of the visual filters, make sure you've the compact filter defined for the same property and move it as described above. This changes the sequence of the property paths in `UI.SelectionFields` and the sequence of the visual filter is adjusted accordingly.

Deleting Filter Fields

To delete a field within the list of the **Filter Fields** perform the following steps:

1. Select a required filter field.
2. Press **trash** icon to delete.

ℹ Note

`Common.Label` annotation isn't deleted along with the filter field, as it can be also used elsewhere in the application.

ℹ Note

When deleting a visual filter, only the respective configuration in `manifest.json` is removed. To remove the respective annotations from the local file, use the cleanup button.

Maintaining Filter Fields Properties

The following **Filter Fields** and **Compact Filter** properties are editable:

- Label Maintenance
- Text
- Text Arrangement
- Display Type

See Appendix for more information on editing Text, Text Arrangement, and Display Type.

Label

To change the label, perform the following steps:

1. Click on the **Filter Fields** in the outline to display its properties in the properties pane.
2. In the **Label** field, add new text.

Removing the label text won't delete any `@title` and `@Common.Label` annotations defined for that property in the upper and lower layers.

ℹ Note

Changing the filter label has a global effect and will influence all occurrences of that field in the application unless it's overridden there.

The following **Visual Filter** properties are editable:

- Measure Label
- Dimension Label
- Dimension Text
- Dimension Text Arrangement
- Measures and Currencies
- Scale Factor
- Number of Fractional Digits
- Sort Order

See Appendix for more information on editing Measures and Currencies.

ℹ Note

Measure and dimension labels as well as a scale factor and Unit of Measure or currency the impact the display of the visual filter title in the following order: **Measure Label** by **Dimension Label** in **Scale factor Measure or Currency Unit** .

ℹ Note

Text values for **Dimensions** should be from the same entity as dimension.

Measure and Dimension Labels

To change the label, perform the following steps:

1. Choose the **Filter** in the outline to display its properties in the **Property Panel**.
2. In the **Measures** or **Dimensions** table, change the value in the **Label** filed.

If you don't define a label, property name for the respective measure and dimension is displayed in the visual filter title.

ℹ Note

Changing the dimension label as well as a label of custom aggregated measure updates the `Common.Label` or `@title` annotation applied to the property used as measure or dimension. Changing the label for the measure built with transformation aggregation, updates the `Common.Label`l annotation applied to `Analytics.AggregatedProperty`.

Scale Factor

By default, the scale factor for the visual filter measure data is calculated automatically based on the data. However, you can explicitly set the desired scale factor by choosing one of the values provided in the drop-down box. If you want to use the

calculated scaling factor, choose **None**.

> **Note**
>
> Scaling factor is defined in `UI.DataPoint` annotation referenced in `UI.Chart` annotation of the visual filter.

Number of Fractional Digits

If the measure data in your visual filter is of numeric type, you can choose how many decimal places to display for it. By default, no decimals are displayed, but you can set it to 1 or 2. For a currency-based measure, the number of decimal places as specified here's only considered if the scale factor is defined. Otherwise, the number of decimal places is based on the currency.

Sort Order

You can sort the measure data in the visual filters represented by the bar chart as follows:

1. Choose Add Sort Property button. The property used for chart measure is set as sort property with ascending direction.
2. Choose Descending in the direction field to sort the measure data in descending order.

> **Note**
>
> You can't sort the chart data in the visual filters represented by the bar chart by different property.

> **Note**
>
> You can't sort the chart data in visual filters based on the line chart.

Table

Table is an essential part of the **List Report** age. It's based on the `UI.LineItem` annotation that can be generated along with the application, unless you choose differently in the SAP Fiori application generator. For more information, see Overview.**Page Editor** lets you further configure the list report table by adding columns and actions.

Currently supported elements for **List Report** table are:

- Table Actions
- Table Columns

Table Sorting

Sorting table data is set in the `Presentation Variant` property of the table. A table node in **Page Editor** layout tree therefore has the annotation based properties `Presentation Variant` and `Sort Order`..

> **Note**
>
> Maintaining `Presentation Variant` and `Sort Order` in analytical charts is described in Analytical Chart.

Presentation Variant

`Presentation Variant` property shows the `UI.SelectionPresentationVariant` or `UI.PresentationVariant` annotation defining the table presentation options, such as sorting or grouping. If *Presentation Variant* is not yet set for the

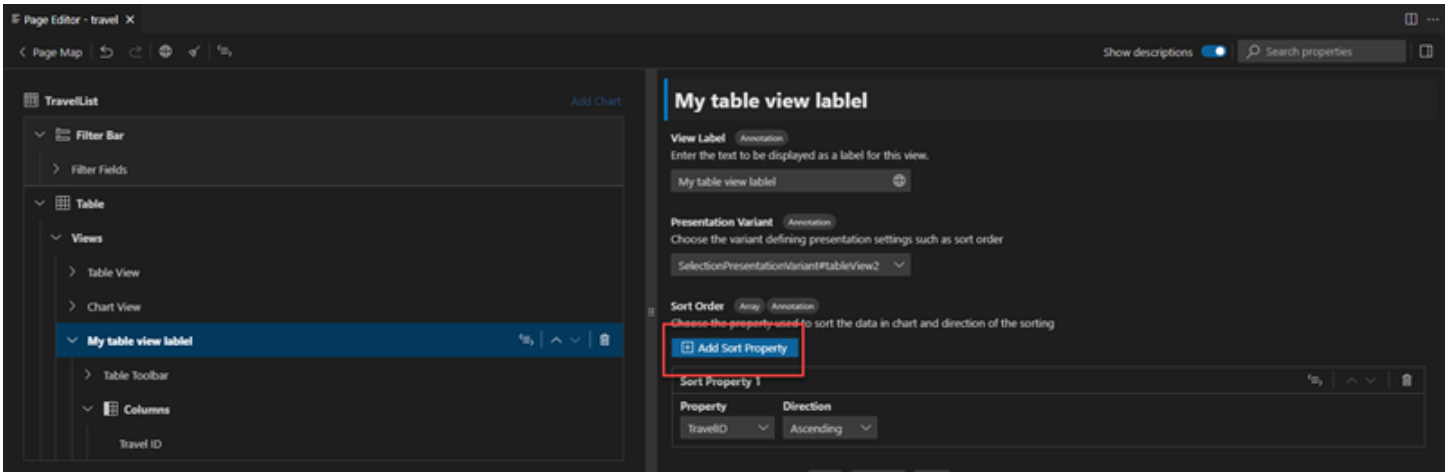
table, you can have it generated by choosing `New` option in this property.

ℹ Note

If your List Report is configured with the Analytical Chart, you can also choose to reuse the `Presentation Variant` applied for that chart. In this case, sort order will apply for both chart and table.

When the *Presentation Variant* is set, you can define one or more direct properties of the entity to sort the table data by. For this, press the Add Sort Property button. A new table row for the sort property is added with the **Property** and **Direction** fields. Update the default values for these fields in the table row if needed.

If you have more than one sort property, you can define in which order they apply to the table data by moving them up and down within the Sort Order property.



You can detach the *Presentation Variant* generated by the Page Editor by setting it to **None** unless your List Report is configured with Multiple Views.

ℹ Note

This action deletes respective `UI.SelectionPresentationVariant` from the manifest.

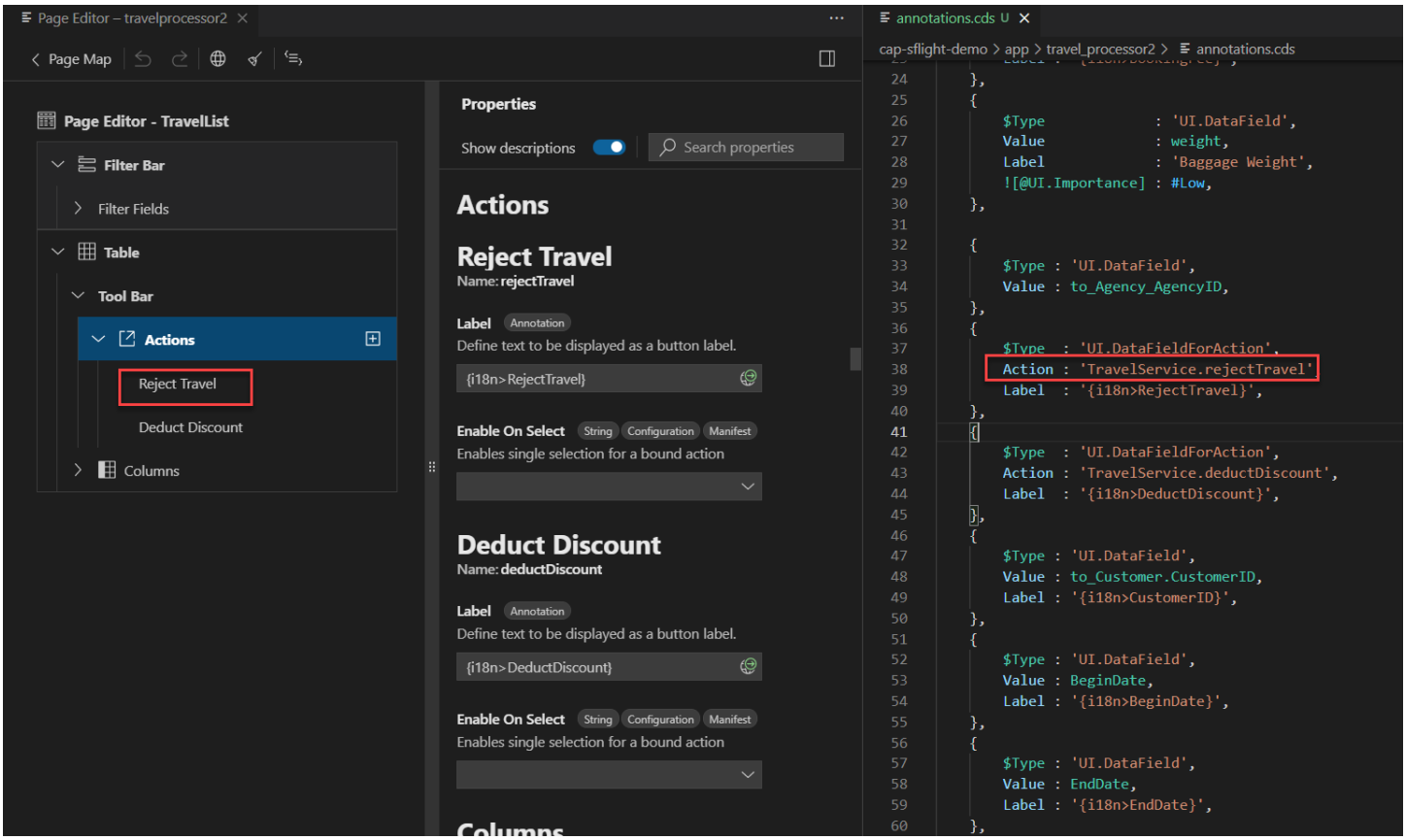
ℹ Note

To remove unreferenced `UI.SelectionPresentationVariant` annotations generated by the Page Editor from the annotation file, run the cleanup procedure that deletes the unreferenced annotation. This will also remove the other unreferenced `UI.SelectionPresentationVariant` and `UI.PresentationVariant` annotations defined with qualifier to keep your annotation file as clean as possible.

Table Actions

Table actions can be placed in a toolbar or inline in table rows in an Object Page section, or a List Report. They're based on the records type `UI.DataFieldForAction` contained in the `UI.LineItem` annotation.

Toolbar and inline action definition differs in the property `Inline` of `UI.DataFieldForAction`. This means, actions displayed as a table columns are defined with the property `Inline` set to `true` in the corresponding `UI.DataFieldForAction`.



Add Action

Creating a table action is possible if there's an available **Action** or **Function** defined in the service. Bound actions and bound functions are only considered if it is applied to the same entity type as `UI.LineItem` defining the table for which the action should be created.

Creating a table action can be applied to the following nodes in the Page Editor:

- List Report: Columns or Toolbar node in a Table node.
- Object Page: Columns or Toolbar node for a Table section.

When action is added as a table column, the `Inline` property of `UI.DataFieldForAction` is added and set to `true`.

Label

The label of a table action, which is shown in the Page Editor is derived from existing annotations `Common.Label` or `@title`(CAP CDS) targeting the action.

If these annotations are missing, the property `Label` is generated within `UI.DataFieldForAction`. You can change the label, the changed value is persisted in `UI.DataFieldForAction`.

ⓘ Note

When an action is deleted, annotations `Common.Label` or `@title` aren't deleted, only the Label property of the `UI.DataFieldForAction` gets deleted along with the table action.

Move Table Action

Moving a table action from **toolbar** to **columns** or vice versa will switch the **Inline** property accordingly. Table action can only be moved within the same table.

Table Columns

You can add columns of different types to the table Table or a Table Section in the **Object Page**.

To add a table column, press + in the columns node and choose the desired column type. Then enter the requested information and choose Add.

ℹ Note

The requested information depends on the selected column type.

Once the table columns are added, you can move them within the table, delete or define additional properties for them in the properties pane. Some properties, such as **Label** and **Importance** can be defined for all table columns, others are specific to the column type. You can see a column type in the properties pane within the column header. For basic columns, the value type, such as **String** or **Decimal** is displayed. For others, its visualization, such as **Chart** or **Rating**.

The following column types are supported:

- Basic Column
- Chart Column
- Rating Column
- Progress Column

Maintain Column Properties

The remaining properties depend on the column type (Chart, Rating, etc) and value type (String, Date, Decimal, etc) you select in the outline.

ℹ Note

You can check the column type in the column header in the **Property Panel**. For basic columns, you see the value type, for others - the specific column type, such as Chart or Rating.

All the properties are by default accompanied with the short descriptions helping you understand and configure them. This documentation provides additional information for the relatively complex properties.

When adding a column to a table the following can be configured for the columns of all types in the **Property Panel**:

Importance

- When you just add a column to the table, the importance is set to **None**. This value means that no importance is specified for the current column. You can change the importance value for table columns to indicate which of these fields should be displayed on the smaller screens such as that of smartphone or tablet.
    - If the column importance value is defined in `UI.LineItem` residing in a lower layer (base layer), the corresponding value is displayed in the drop-down menu with a suffix (base layer). Example: High (base layer).

- If the Importance value is changed to **None**, the `UI.Importance` annotation will be deleted from the local annotation file.

ℹ Note

Edit in source code icon is provided also when the **Importance** is set to **None**. You can use it to navigate to the *UI.DataField* record representing the table column where `UI.Importance` annotation is usually defined.

Label

When a column is added to a table, the **Page Editor** checks for the *Common.Label* and `@title` annotations on a property used as column value.

ℹ Note

`@title` annotation is only relevant and checked for in the CAP projects.

- If these annotations are provided, `UI.DataField` record is generated without the **Label** property. In this case the value of `Common.Label` or *@title* property is used as a column title and displayed in the **Label** field for the column in the properties pane.

  ### ℹ Note

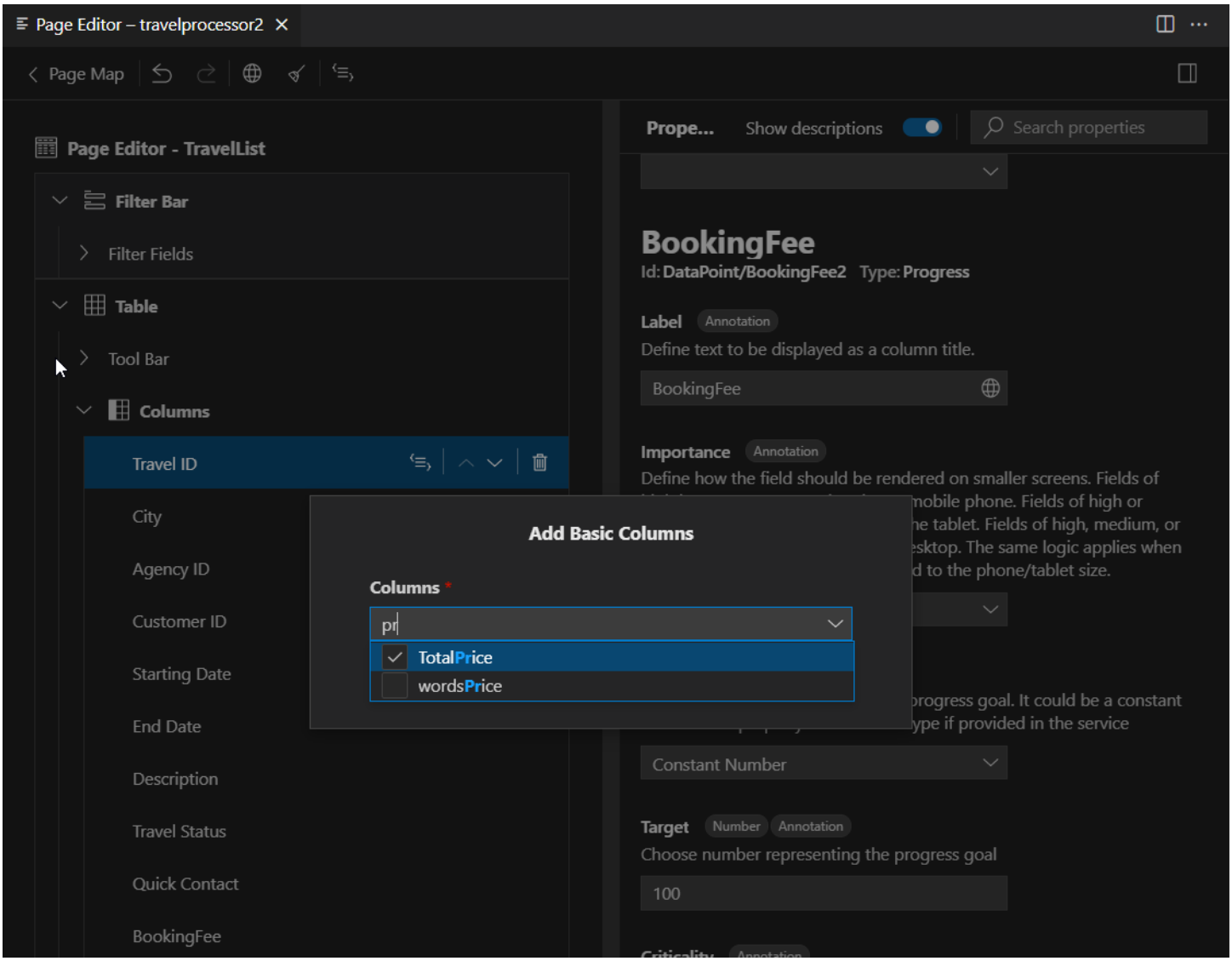
  If you change this value in the properties pane, `Common.Label` or `@title` annotations will not be affected to avoid the unexpected changes in other parts of the application. Instead the the `Label` property with a new value will be generated in the respective `UI.DataField` record and thus the change will be specific to the column title of the modified table.

- If these annotations are not yet provided, `UI.DataField` record is added with the auto-generated label. You can then change this label for the given context in the properties pane.

During deletion of the column, the annotations `Common.Label` or `@title` are not removed. Only those labels that are directly maintained in a record are deleted as the record is completely removed.

Basic Column

Basic column is used for the standard representation of the value type. For example, it shows string values as text and numeric values as numbers. You can add multiple basic columns at a time by selecting more than one value in the **Columns** field. To easier find the desired value, you can filter the list of available options by typing in a few characters of the desired value name.



Adding a Basic Column

To add a basic column to a table to a section, perform the following steps:

1. Click `Add Basic Column` when choosing + button in Columns node in the Page Editor .
2. Select Columns via a tree control.
3. Click Add.

> **Note**
> You cannot add the column based on the same value twice into the table.

Column properties, can be configured in the Property Panel such as:

- Label
- Importance
- Text and Text Arrangement (for all value types exept boolean)
- Display Type (for string values)
- Criticality and Criticality Representation(for string and numeric values)
- Measures and Currencies (for numeric values)

Please see Column Properties for information on defining **Label** and **Importance**. Please see Appendix for information on defining and editing the remaining properties.

Moving Basic Column

To move a column within a table, use one of the following options:

- **Drag-and-drop**

  Hover over the table column in outline, press and hold the mouse button while moving the mouse pointer to the different position within the table. Release the mouse button at the desired position. Eligible positions are highlighted in green.

  With drag-and-drop option you can move multiple columns at once by pressing CTRL+.

- **Arrow Buttons**

  Press Move up or Move down buttons next to the column name. This option only moves one column at a time.

Deleting Basic Column

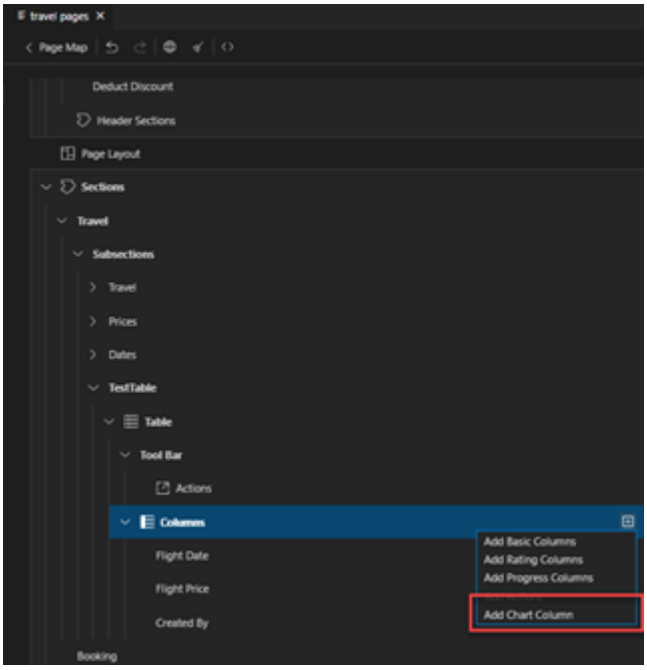
To delete a column in the application, perform the following steps:

1. Navigate to a column.
2. Click Delete icon.

   The **Delete Confirmation** pop-up window appears.
3. Click Delete to confirm the action.

Chart Column

Chart column can be added to a table that is part of **List Report** or in an **Object Page** section.



Depending on the desired chart type, you need to choose the values for the mandatory properties.

Add a Chart Column

To add a chart column to a table to a section, perform the following steps:

1. Click `Add Chart Column` when choosing [+] button in Columns node in the **Page Editor** .

   > **Note**
   >
   > If `Add Chart Column` option is disabled if the table entity doesn't have any numeric properties.

2. Select **Chart Type** via a tree control.
3. Click [Add]. Depending on the desired chart type, you need to choose the values for the mandatory properties.
   - When the chart column is added, a new `UI.Chart` and `UI.DataPoint` annotation is created.

Column properties, can be configured in the **Property Panel**.

See Column Properties and Appendix for information on defining and editing the properties.

Chart type: Bullet

- **Value** numeric property to represent the chart data.
- **Maximum Value (Path)** fixed number to represent the maximum possible value in the chart.

> **Note**
>
> You can set the Maximum Value to the numeric property in the Properties pane once you add the chart column.

Chart type: Radial

- **Value** numeric property to represent the chart data.
- **Target Value (Path)** numeric property to represent the maximum possible value in the chart.

When the Chart Column is added, a new `UI.Chart` and `UI.DataPoint` annotation is created.

> **Note**
>
> The generated chart is based on the minimum required properties entered when adding the chart column. You can configure it further in the **Property Panel** by defining additional properties for the selected chart type, such as criticality, thresholds, etc.

Move Chart Column

To move a column within a table, use one of the following options:

- **Drag-and-drop**

  Hover over the table column in outline, press and hold the mouse button while moving the mouse pointer to the different position within the table. Release the mouse button at the desired position. Eligible positions are highlighted in green.

  With drag-and-drop option you can move multiple columns at once by pressing CTRL+.

- **Arrow Buttons**

  Press [Move up] or [Move down] buttons next to the column name. This option only moves one column at a time.

Delete Chart Column

To delete a column in the application, perform the following steps:

1. Navigate to a column.
2. Click Delete icon.

   The **Delete Confirmation** pop-up window appears.
3. Click Delete to confirm the action.

Rating Column

Rating column can be added to a **List Report** table or an **Object Page** section.

Adding a Rating Column

To add a rating column to a table to a section, perform the following steps:

1. Click `Add Rating Columns` when choosing + button in Columns node in the **Page Editor** .
2. Select **Columns** via a tree control.
3. Click Add, a new `UI.DataPoint` annotation is created with the following values:
   - `Value` property is set to the property chosen by the user.
   - `TargetValue` property is set to 5.
   - `Visualization` property is set to enum value **Rating**.

Column properties, can be configured in the **Property Panel**.

Please see Column Properties and Appendix for information on defining and editing the properties.

Moving Rating Column

To move a column within a table, use one of the following options:

- **Drag-and-drop**

  Hover over the table column in outline, press and hold the mouse button while moving the mouse pointer to the different position within the table. Release the mouse button at the desired position. Eligible positions are highlighted in green.

  With drag-and-drop option you can move multiple columns at once by pressing CTRL+.
- **Arrow Buttons**

  Press Move up or Move down buttons next to the column name. This option only moves one column at a time.

Deleting Rating Column

1. Navigate to a column.
2. Click Delete icon.

   The **Delete Confirmation** pop-up window appears.

3. Click Delete to confirm the action.

Progress Column

Progress Indicator column can be added to a **List Report** table or an **Object Page** section.

Adding a Progress Column

To add a progress column to a table to a section, perform the following steps:

1. Click `Add Progress Column` when choosing + button in Columns node in the **Page Editor** .
2. Select **Columns** via a tree control.
3. Click Add, a new `UI.DataPoint` annotation is created created with the following values:
    - `Value` property is set to the property chosen by the user.
    - `TargetValue` property is set to 100 by default.
    - `Visualization` property is set to enum value **Progress**.
    - `UI.LineItem` is updated with a new `UI.DataFieldForAnnotation` containing the reference to this `UI.DataPoint` and a `Label` property generated based on the selected value.

> **Note**
>
> Add progress column option is disabled if the table entity and associated entities do not have any numeric properties or all of them are already used in the table.

Column properties, can be configured in the **Property Panel**.

Please see Column Properties and Appendix for information on defining and editing the properties.

Moving Progress Column

To move a column within a table, use one of the following options:

- **Drag-and-drop**

  Hover over the table column in outline, press and hold the mouse button while moving the mouse pointer to the different position within the table. Release the mouse button at the desired position. Eligible positions are highlighted in green.

  With drag-and-drop option you can move multiple columns at once by pressing CTRL+.

- **Arrow Buttons**

  Press Move up or Move down buttons next to the column name. This option only moves one column at a time.

Deleting Progress Column

To delete a column in the application, perform the following steps:

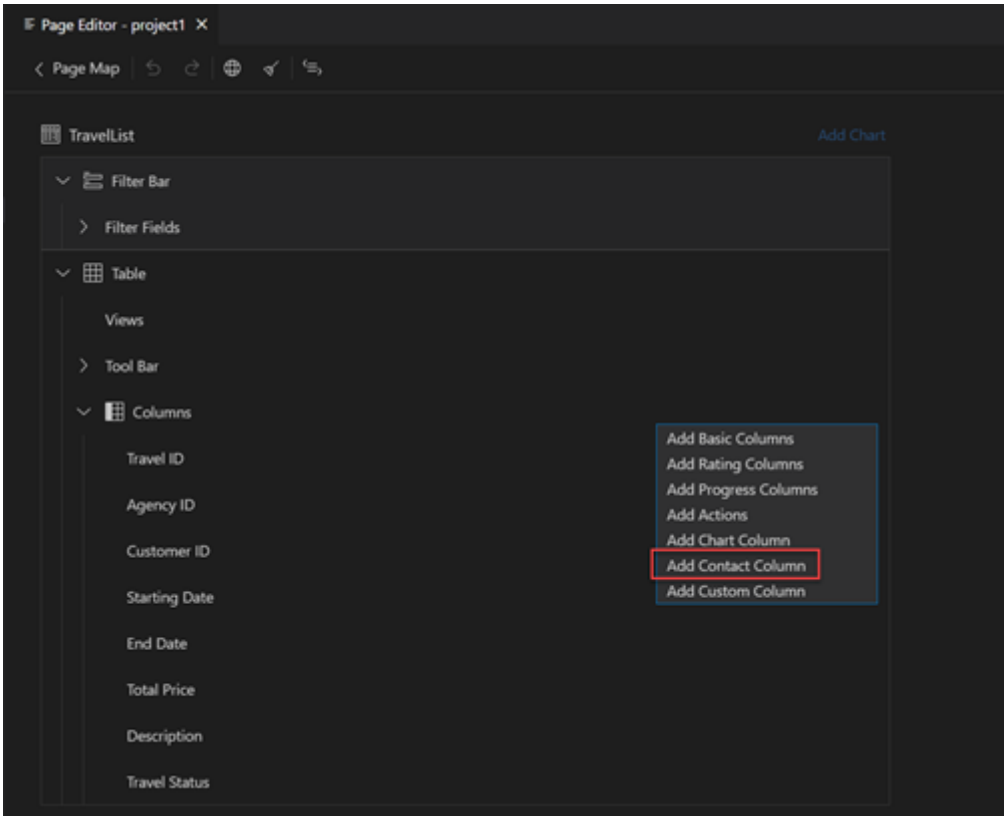
1. Navigate to a column.
2. Click Delete icon.

The **Delete Confirmation** pop-up window appears.

3. Click Delete to confirm the action.

Contact Column

Contact column can be added to a table that is part of **List Report** or in an **Object Page** section.



Add a Contact Column

To add a contact column to a table to a section, perform the following steps:

1. Click Add Contact Column when choosing + button in Columns node in the **Page Editor** .
2. Select **Contacts** via a tree control.
3. Click Add. A new `Communication.Contact` annotation is created with Contact Name label by default. You can change the default label in the **Property Panel**.

Column properties, can be configured in the **Property Panel**.

See Column Properties and Appendix for information on defining and editing the properties.

Move a Contact Column

To move a column within a table, use one of the following options:

- **Drag-and-drop**

  Hover over the table column in outline, press and hold the mouse button while moving the mouse pointer to the different position within the table. Release the mouse button at the desired position. Eligible positions are highlighted in green.

With drag-and-drop option you can move multiple columns at once by pressing CTRL+.

- **Arrow Buttons**

  Press Move up or Move down buttons next to the column name. This option only moves one column at a time.

Delete a Contact Column

To delete a column in the application, perform the following steps:

1. Navigate to a column.
2. Click Delete icon.

   The **Delete Confirmation** pop-up window appears.
3. Click Delete to confirm the action.

Maintain Contact Column Properties

A contact column or field has the following properties.

Label

Label for Contact Field or Contact Column have the same behavior as for a regular Field or Column, see Maintain Column Properties.

Importance

Importance for Contact Field or Contact Column have the same behavior as for a regular Field or Column, see Maintain Column Properties.

Contact

Property **Contact** represents property `fn` of `Communication.Contact`. You can't change the property used as **Contact Name** in the **Property Panel**. If you need to have your Contact column based on a different property, delete this column and use the + button to add a new contact column.

Job Title

Property **Job Title** represents property title of `Communication.Contact`.

Photo

Property **Photo** represents property photo of `Communication.Contact`.

Role

Property **Role** represents property role of `Communication.Contact`.

Address

Property **Address** represents collection property `adr` of `Communication.Contact`, which has record type `Communication.AddressType`. You can maintain multiple addresses, each represented by a table row.

The row contains the following fields:

- **Street** - representing property street of `Communication.AddressType`
- **City** - representing property locality of `Communication.AddressType`
- **State/Province** - representing property region of `Communication.AddressType`
- **Postal Code** - representing property code of `Communication.AddressType`
- **Country** - representing property country of `Communication.AddressType`

Phone

Property **Phone** represents collection property `tel` of `Communication.Contact`, which has record type `Communication.PhoneNumberType`. User can maintain multiple phone entries, each represented by a table row.

The row contains the following fields:

- **Phone** - representing property `uri` of `Communication.PhoneNumberType`
- **Type** - with options `Work` (default), `Mobile`, `Fax`, depending on if `type` of `Communication.PhoneNumberType` contains enum value `work`, `cell` or `fax` respectively.

Email

Property **Email** represents collection property `email` of `Communication.Contact`, which has record type `Communication.EmailAddressType`. User can maintain multiple email entries, each represented by a table row.

The row contains the following field:

- **Email** - representing property email of `Communication.EmailAddressType`

Multiple Views

You can configure your **List Report** to display one or more additional tables and charts next to the main **List Report** table in separate views. The user of your application can switch between views using an icon tab bar. The tables in the views can be based on any entity in your service. The charts can be based on any entity as long as it contains aggregatable and groupable properties.

ℹ Note

Groupable and aggregatable properties are defined on the service level with `@Aggregation.ApplySupported`. If it'sn't provided, you can't generate chart views with **Page Editor**. If you want to use custom aggregations for chart measures, your service should also have properties aggregated with `@Aggregation.CustomAggregate`.

If you want to use the transformation aggregations, make sure your app runs with SAPUI5 version **1.106** or **higher** to ensure transformation aggregation with `@Analytics.AggregatedProperty` is supported. Transformation aggregation with `@Analytics.AggregatedProperties` isn't supported as this annotation is deprecated in favor of `@Analytics.AggregatedProperty`, see OData Analytics .

ℹ Note

You can't configure multiple views if your **List Report** is configured to display an **Analytical Chart** above or as alternative to the main **List Report** table, see Analytical Chart. You can delete**Analytical Chart** to enable adding views.

Adding a Table or Chart View

1. In the Page Editor for the List Report page, click + on the Views node.
2. In the popup menu, select Add Table View or Add Chart View. A popup dialog allows to choose an `Entity` from the OData Service. If you chose to Add Chart View, you're also prompted for the minimum required data to generate a chart: chart type, a dimension, and a measure.

   A measure can be specified by selecting one of the following:

   - Use existing measure
   - Create new measure

   If you choose to use existing measure, select one of the available measures defined with custom or transformation aggregations in the Name field.

   If you choose to create new measure, choose the aggregatable property and one of the supported aggregation methods.

   This allows you to create a new dynamic measure and use it in the chart.

   Note

   The technical name and the label is generated automatically . You can then adjust the generated label in the Property Panel.

   Note

   Create new measure only works with transformation aggregations so it should be used for the apps runs with SAPUI5 version **1.106** or **higher**. If all the possible measures based on all the aggregatable properties and supported aggregation methods are already defined in the project, you can't create a new measure. Use the existing measure instead.

3. Press Add in the dialog. In the Page Editor, a new subnode is appended to the Views node with generated view label.

   Note

   The table is added with no columns. You can add columns using + button for the **Columns** subnode.

The following changes take place in the annotation file:

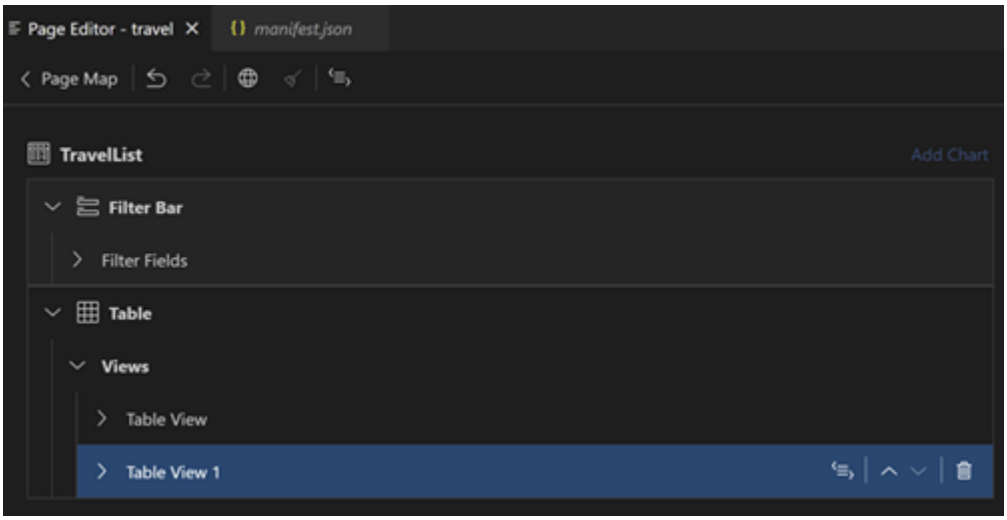
- `UI.LineItem` or `UI.Chart` annotation with qualifier is generated targeting the `EntityType` referenced by the selected `EntitySet`.
- If you chose to create a new measure, `@Analytics.AggregatedProperty` is applied to the selected aggregatable property with your chosen aggregation method.
- `Views/Paths` section in `manifest.json` is generated or appended with an entry pointing to the generated `UI.SelectionPresentationVariant`. If different from the main `EntitySet` of the, List Report the chosen `EntitySet` is added to the paths entry.

Note

If the `Views/Paths` section in `manifest.json` didn't exist before, second entry is created with table or chart view, with the first entry representing the original table.

Moving Table or Chart View

All table or chart views are represented as subnodes of **views** node. Drag and drop them or use the corresponding Move Up + Move Down buttons to change view sequence in Lt Report.



Deleting Table or Chart View

To remove a table or chart view, press the Delete icon on the corresponding **views** node.

ℹ Note

It's not possible to remove the last table view that is based on the main entity set of the **List Report**.

If all views except the last table view are deleted, **List Report** is implicitly converted into plain **List Report**:

- `Views/Paths` section in `manifest.json` is removed.
- If needed: `defaultTemplateAnnotationPath` in `manifest.json` is created to point to an `UI.SelectionPresentationVariant`.
- **Views do**n't display any subnodes: details of the **List Report** (columns, actions, etc.) can be maintained in the `Table` node.

Table or Chart View Properties

View Label

View label is a text used to indicate a view on an icon tab bar above the table or chart. It's auto-generated when you add a view. You can change the generated label by entering the text that is meaningful in your application context.

ℹ Note

View label can be prepared for translation, for more information see Internationalization (i18n)Internationalization (i18n).

Presentation Variant

Presentation Variant property lets you define the representation of the table or chart data such as sorting.

Presentation Variant property lets you sort the table data and is maintained the same way as in the standard **List Report** table. See Table for more information.

Presentation Variant for the chart view is maintained the same way as in the Analytical Chart.

ℹ Note

As different views in the **List Report** aren't necessary based on the same entity, you can't reuse the same *Presentation Variant* for different views. For this reason, the options `From Chart` or `From Table` aren't provided. To have the same representation of different views, just define the same options for each related view.

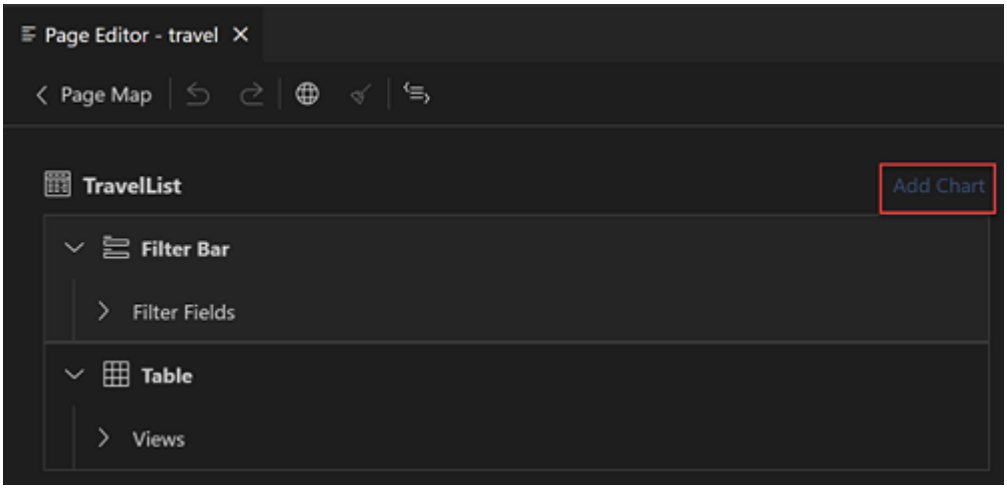
Analytical Chart

In **Page Editor**, you can configure the **List Report** page to display aggregated data of your main entity as an analytical chart above or as an alternative to the list report table. This setup is known in `OData V2` as **Analytical List Page**. In `OData V4`, it's however not a separate template but a flavor of the **List Report** template. You can configure it for your new or existing **List Report** if your main entity contains aggregatable and groupable properties.

Prerequisites

- Your list report doesn't contain Multiple Views.
- Your main entity contains aggregatable and groupable properties.

If the Add Chart button is inactive, hover over the disabled Add Chart button to get a hint on the reason. If aggregated or groupable properties aren't defined, the tooltip informs you what annotations are needed to enable it. If your **List Report** is set up with Multiple Views, the tooltip reminds you of that, and you can delete all the views in your **List Report** except the single table based on main entity to enable the Add Chart button.



Note

Page Editor supports adding charts with measures based on custom and transformation aggregations. If you want to use the transformation aggregations, make sure your app runs with SAPUI5 version **1.106 or higher** to ensure transformation aggregation with `@Analytics.AggregatedProperty` is supported. Transformation aggregation with `@Analytics.AggregatedProperties` isn't supported as this annotation is deprecated in favor of `@Analytics.AggregatedProperty`, see OData Analytics .

Adding Analytical Chart

Perform the following steps to add an analytical chart to a **List Report**.

1. Click Add Chart button in the header of the **Page Editor**.
2. Enter the minimum required data to generate a chart: chart type, a dimension, and a measure.

   A measure can be specified by selecting one of the following:

- Use existing measure
- Create new measure

If you choose to use existing measure, select one of the available measures defined with custom or transformation aggregations in the **Name** field.

If you choose to create new measure, choose the aggregatable property and one of the supported aggregation methods.

This allows you to create a new **dynamic** measure and use it in the chart.

> **Note**
>
> The technical name and the label are generated automatically . You can then adjust the generated label in the **Property Panel**.

3. Press Add.

The respective annotation and manifest changes are generated and basic chart is displayed in your list report above the table.

Deleting Analytical Chart

The **Analytical Chart** can be deleted by pressing the delete icon on the layout node. This reverts the floor plan into a conventional **List Report** with a single table.

Maintain Analytical Chart Properties

When you generate a chart, only required properties are defined. To edit the basic chart properties and define additional ones in the properties pane, choose the **chart** node in the outline and update its properties in the **Properties** pane as follows.

Chart Type

Chart type defines how the aggregated data in your entity are visualized in the application. Based on your data nature and your needs, choose one of the provided chart types to optimally visualize you data in the chart.

Title

Chart title is displayed above the chart. You can enter the free form text briefly describing the data, their relationship, or or purpose of the chart.

> **Note**
>
> Chart title can be prepared for translation, for more informationsee Internationalization (i18n).

Measures

Chart measures are the aggregated properties representing values of the chart. **Page Editor** supports custom aggregations and transformation aggregations.

> **Note**
>
> If you want to use custom aggregations for chart measures, your service should also have properties aggregated with `@Aggregation.CustomAggregate`. If you want to use the transformation aggregations, make sure your app runs with SAPUI5 version **1.106 or higher** to ensure transformation aggregation with `@Analytics.AggregatedProperty` is supported. Transformation aggregation with `@Analytics.AggregatedProperties` isn't supported as this annotation deprecated in favor of `@Analytics.AggregatedProperty`, see OData Analytics .

When generating a chart, you choose just one measure. Afterwards, you can change it, assign a label for it and add additional measures if needed in the **Property Panel**.

Each chart must have at least one measure set as default. It is used for displaying the chart data when the end user starts the application, unless it is defined differently in variant management. All the other measures defined for the entity are available to the end user on demand in chart preferences. The **Measures** property of the chart provides all the measures available for the entity the chart applies to. You can set any of them as default by switching the **Default** property on for the respective measure. You can change the sequence of the default measures.

ℹ Note

If a chart has both custom and transformation-based (dynamic) measures set as default, their sequence can't mixed due to the nature of the `UI.Chart` annotation.

Add Measure

You can add a new measure for the entity, press the Add New Measure button and choose the aggregated property and supported aggregation method for it in the pop-up dialog. When you choose Apply, a new **dynamic** measure is generated for the chart entity and set as default in the chart. The technical name and the label are generated automatically . You can then adjust the generated label in the property panel.

Modify Measure

To change the **Default** property of a measure, set a different measure as default and switch off the **Default** property for the current one.

Define Measure Label

Measure label depends on the `Common.Label` or (in CAP CDS) `@title` annotation applied to the property used as a measure. If it'sn't defined, you can enter the text for it in the **`Label`** property displayed in the **Measure** row next to the **Property**. If it's already defined, you can update it. Removing the label text won't delete any `@title` and `Common.Label` annotations defined for that property in the upper and lower layers.

ℹ Note

Changing the measure label has a global effect and will influence all occurrences of that field in the application unless it's overridden there.

ℹ Note

Measure label can be prepared for translation, for more information see Internationalization (i18n).

Add, Move Measures

You can add additional measures to your chart if more than one direct property of the main entity is annotated as aggregatable. For that, press the Add Measure button and choose the desired property.

ℹ Note

You can't add the same measure to the chart twice. If all the aggregatable properties are already used as chart measures, Add Measure button is dabled.

You can change the sequence in which default measures are displayed in the **Analytical Chart**. For the, drag and drop the measure rows within the **Measures** property or use the Move Up/Move Down icons in the measure row header.

Delete Measures

You can delete any transformation-based measure as long as it's defined for the current app and at least one measure remains default for the chart. For this, press the Delete icon in the measure row header

Dimensions

Chart dimensions are groupable properties categorizing the measures in the chart. When generating a chart, you choose just one dimension to be used by default. Afterwards, you can change it, assign a label and set additional dimensions as default if needed in the **Property Panel**. Dimensions property lists all the dimensions available for the entity the chart is applied to.

Each chart must have at least one default dimension. It's used for categorizing the chart data when the end user starts the application, unless it defined differently in variant management. All the other dimensions defined for the entity are available to the end user on demand in chart preferences. The **Dimensions** property of the chart provides all the measures available for the entity the chart applies to. You can set any of them as default by switching on the **Default** property for the respective measure. You can change the sequence of the default measures.

Modify Dimension

To change the property used as a dimension, choose a different groupable property in the **Property** drop-down. To change the dimensions used by default, use the **Default** switch in the header of the respective dimension rows.

Define Dimension Label- Dimension label depends on the `Common.Label` or (in CAP CDS) `@title` annotation applied to the property used as a dimension. If it'sn't defined, you can enter the text for it in the **Label** property displayed in the **Dimension** row next to the **Property**. If it's already defined, you can update it. Removing the label text won't delete any `@title` and `Common.Label` annotations defined for that property in the upper and lower layers.

> **Note**
>
> Changing the dimension label has a global effect and influences all occurrences of that field in the application unless it's overridden there.

> **Note**
>
> Dimension label can be prepared for translation, for more information see Internationalization (i18n).

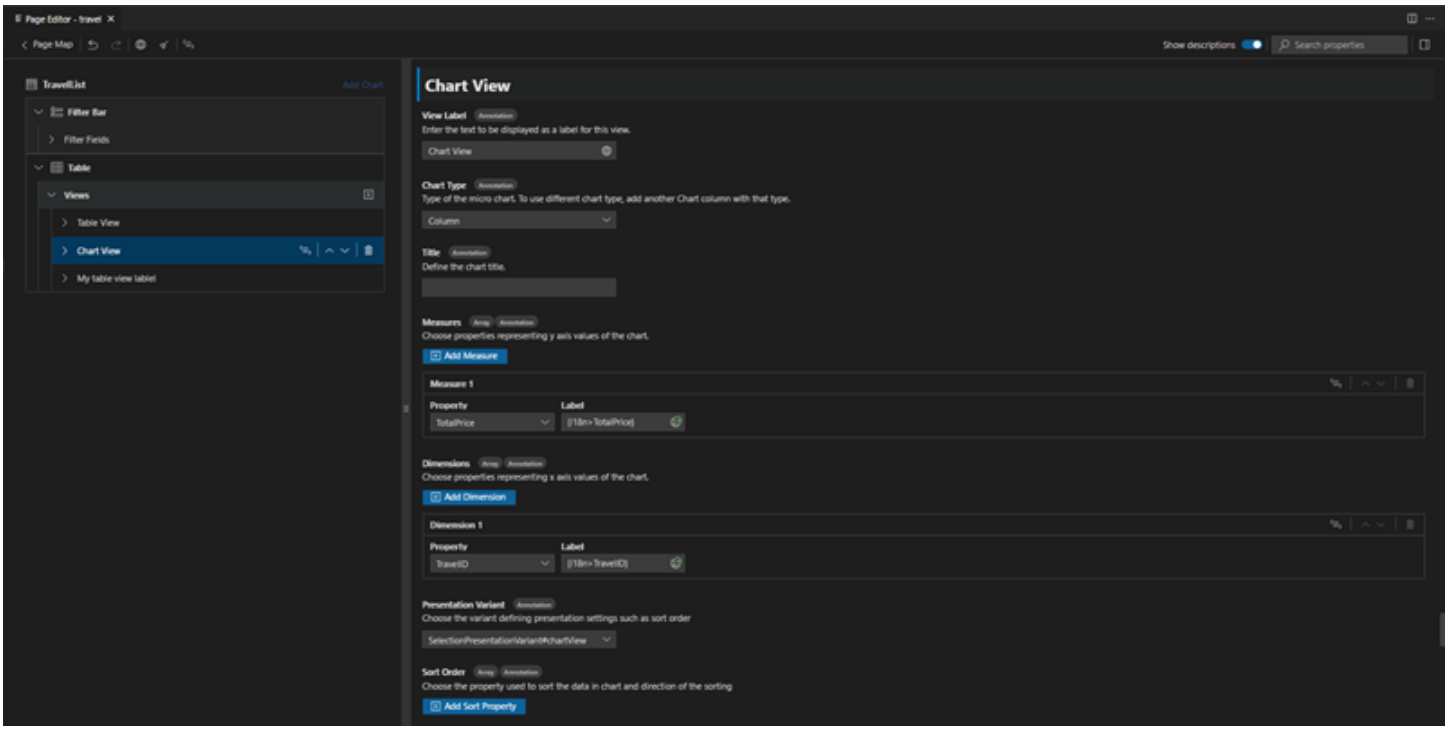
Set Dimension Text and Text Arrangement

You can set the Text and Text Arrangement for the dimension values in the respective **Dimension** table. For more information, on setting Text and Text Arrangement, see Appendix.

> **Note**
>
> Text values for **Dimensions** should be from the same entity as dimension.

Move Dimension

You can change the sequence in which measures are grouped by dimensions in the analytical chart. For this, drag and drop the default dimension rows within the **Dimensions** property or use the Move Up/Move Down icons in the dimension row header.



Presentation Variant

`Presentation Variant` property is used to sort the chart data. It shows the `UI.SelectionPresentationVariant` or `UI.PresentationVariant` annotation defining that order. If `Presentation Variant` isn't yet set for the chart, you can have it generated by choosing `New` option in this property. You can also reuse the `Presentation Variant` applied for the list report table by choosing the From Table option. In this case, sort order applies for both chart and table.

Sort Order

When the `Presentation Variant` is set, you can define one or more properties to sort the chart data by. For this, press the Add Sort Property button then choose one of the direct properties of the chart entity to sort by and the sort direction. If you've more than one sort property, you can define in which order they apply to the chart data by moving them up and down within the Sort Order property. To move the properties, drag and drop the property rows within the Sort Order property or use the Move Up/Move Down icons in the row header.

You can change the properties used for sorting, update the sorting direction as well as delete one or more sorting properties. You can remove the `Presentation Variant` applying to the chart by setting it to **None**. In this case, the `UI.Chart` annotation defining an analytical chart is referenced in manifest directly and sorting won't be applied.

ℹ Note

This action deletes respective `UI.SelectionPresentationVariant` or `UI.PresentationVariant` from the manifest.

ℹ Note

To remove unreferenced `UI.SelectionPresentationVariant` or `UI.PresentationVariant` annotations from the annotation file, run the cleanup procedure that deletes the unreferenced annotation. You can always generate a new `Presentation Variant` or use the one defined for the table if any by choosing the `New` and From Table options respectively. You can generate a new`Presentation Variant`.

Form and Object Page

The following links give you general information about **Form and Object Page**:

- Fiori Design Guidelines.
- Developing Low-Code Business Application.

The current supported elements of the **Form and Object Page**:

- Header
- Form Section
- Table Section
- Identification Section
- Group Section
- Footer

Header

Header Properties

The following annotation-based properties can be defined on the Header node of an **Object Page**

- Type Name
- Type Name Plural
- Title
- Description
- Image
- Initials
- Icon URL

All properties are based on annotation `@UI.HeaderInfo`.

If `@UI.HeaderInfo` doesn't exist, it will be created as soon as one of the properties above gets a value.

If `@UI.HeaderInfo` annotation it's defined in the lower layer, such as service, the values of these properties are marked with the "(base layer)" suffix indicating the value origin. Once changed at least one property value, the complete annotation is copied to the local annotation file and (base layer) suffix is no longer displayed to indicate it.

Type Name/Type Name Plural

String properties describing the main object of the page. *Type Name* is displayed in on the very top of the **Object Page**: *Type Name Plural* represents a plural form of the object name and is displayed as a table header on the previous page. As these properties are mandatory, they're set to the empty string if not (yet) defined otherwise. The properties support internationalization. See Internationalization (i18n) for more information.

Title

Property representing the main object of the page. It's displayed in the page header area. You can choose one of the direct properties of the page entity provided in the drop-down box. If you set it to **None**, **Object Page** header will not contain the title.

Default text will be displayed instead. Always define the **Title** if the property **Visible** of the page header is set to **true**.

ⓘ Note

None option isn't available if the Title is defined in a lower layer such as service.

Image

- Adds property `ImageUrl` with the selected property as a value to the `UI.HeaderInfo`.
- Value is a path pointing to string properties of the entity or of a to one associated entity.
- To remove `ImageUrl` property, you can select option **None**.

For more information on images, see: Using Images, Initials, and Icons.

Initials

- Adds property `Initials` with the selected property as a value to the `UI.HeaderInfo`.
- Value is a path pointing to string properties of the entity or of a to one associated entity.
- To remove `Initials` property, you can select option *None*

Icon URL

- Adds property `TypeImageUrl` with the sap icon text as a value to the `UI.HeaderInfo`.

  ### Example

  sap-icon://accept

- Value is a string pointing to sap icon, for example, from icon explorer.

Header Section

Header sections show that the key information on the **Object Page** entity are displayed in the header area. The visualization of this information depends on the *section type. information*.

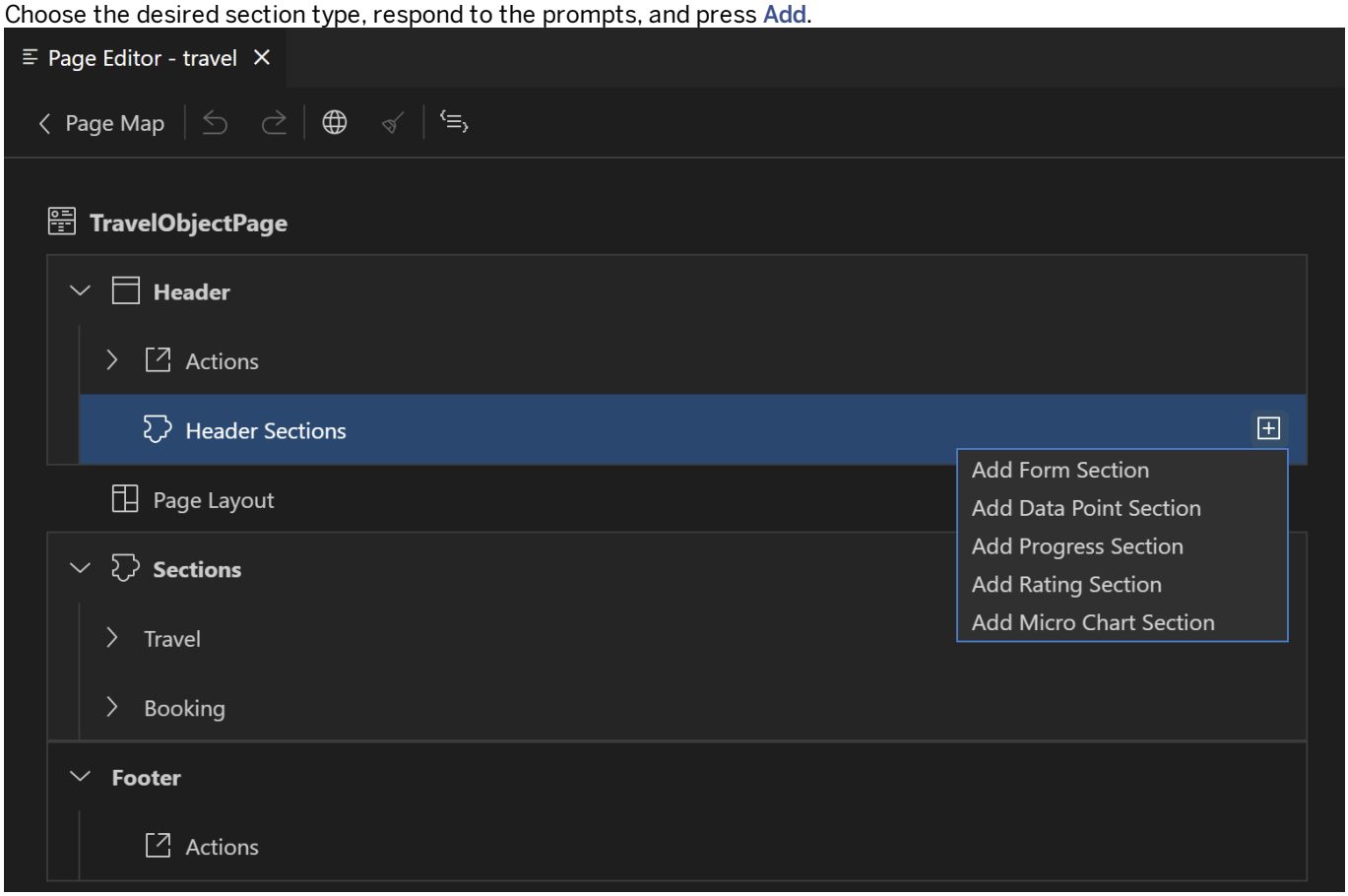
Add Header Section

To add a Header section, perform the following steps:

1. Click the **Object Page** to open the **Page Editor**.
2. Navigate to the Header section now and click + icon.

   As a result, a list of section types supported in the page header appears.

3. Choose the desired section type, respond to the prompts, and press **Add**.



4. Depending on the section type selected, additional information is needed:
    - Form Section - Label
    - Data Point Section - Value Source Property
    - Progress Section - Value Source Property
    - Rating Section - Value Source Property
    - Chart Header Section - Chart Type. You're prompted for more information depending on the selected chart type, same as for Chart Column.

Once the header section is generated, you can add and maintain its properties in the **Property Panel**.

Form Section

The Form header section displays a group of fields under the common label. Once you add the Form section, no fields are added automatically. Add the fields you need using the [+] icon in the **Fields** node.

For more information, see Form Section

Progress Section

The **Progress** header section visualizes the numeric value you chose as an indicator of a progress towards a certain target. You can modify the generated label and a default target as well as define a description for your progress indicator, apply the semantic coloring based on the value criticality and provide a tooltip.

Target initially the progress indicator is generated based on the value you entered and the default target (goal) of 100. You can then modify the target by setting it to a different numeric constant or choose a numeric service property that represents a target. For that, you first choose the target value type and then the desired number or property.

See Appendix for more information on Criticality, Measures and Currencies and Tooltip.

Data Point Section

The **Data Point** header section is used to display the single point of the key data. It's typically a number but can also be textual, for example, a status value. Initially, it is generated with a minimum property based on the value you entered. You can then enhance it in the properties pane with additional features, such as semantic coloring based on criticality. You can also add a tooltip describing the value. If your data point represents a numeric value, you can additionally define the measure or currency for it if this isn't done in the base level.

See Appendix for more information on Criticality, Measures and Currencies and Tooltip.

Rating Section

The **Rating** header section displays numeric value you chose with the corresponding number of stars out of the certain maximum (target). Initially it's generated based on the value you entered and default target value of 5. Subsequently, you can modify the generated label and set the target to any other integer number in the properties pane as well as enter a description and a tooltip.

See Appendix for more information on Criticality, Measures and Currencies and Tooltip.

Micro Chart Section

The **Micro Chart** header section allows you visualizing the numeric properties of your service as micro charts of different types. Initially micro charts are generated based on the minimum required information you entered and some assumed defaults. You can modify some of the generated chart properties as well as define optional ones in the properties pane.

Required and optional properties you can configure depend on the selected chart type.

> **Note**
>
> You can't change the type or main value (measure) of the micro chart. If you need to modify one of these properties, just add a new micro chart section and delete an existing one instead.

Radial Chart

Radial chart displays the numeric value compared with the target. Both values you choose when generating a radial micro chart. Then you can choose a different numeric service property to be used as a target in the properties pane, as well as set a description for your chart and apply the semantic coloring based on the value criticality.

See Appendix for more information on Description and Criticality.

Bullet Chart

Bullet chart visualizes the numeric value on a given scale. Besides, the main value, you can display addition ones, such as target or forecast on the same scale as well as set a description for your chart and apply the semantic coloring based on the value criticality.

Target Value

Apart from the main value, bullet chart can also display an additional value as a target. You can choose a numeric service property to be used as a target in the properties pane.

Forecast Value

Bullet chart can also display an additional value to indicate a forecast. You can choose a numeric service property to be used as a forecast value in the properties pane.

Minimum and Maximum Values

Bullet Chart scale is based on the minimum value (by default 0) and maximum value you chose when generating this micro chart.

You can modify the scale of the chart by updating these minimum and maximum values. You can either define these values based on the service properties of numeric type or as fixed number. For that, you first choose the Maximum (Minimum) value type and then the desired number or property.

Criticality

To semantically color the bullet chart according to the value criticality, you can either choose a property with the criticality information or define the criticality information in place. So, first you need to choose the Criticality Source. When set to Property, you're prompted to choose a service property containing the criticality information. When set to Calculation, you have to choose the desired improvement direction:

- Minimize: low values are considered as best
- Maximize: hight values are considered as best
- Target: values close to the target are considered as best Depending on the selected direction, enter the values to be used as threshold for critical and warning coloring.

See Appendix for information on setting the radial chart description.

Move Header Section

The user can change the order of the sections in the application header. By using the drag-and-drop functionality, drag the required section to a different position within the **Header Sections** node:

- When dropped, the records in the `UI.Facets` collection are reordered.
- When SAP Fiori application is rendered, sections are displayed based on the records sequence in the `UI.HeaderFacets` annotation.

Move multiple sections

To move the multiple sections to another position, perform the following steps:

1. Use the Ctrl + Click combination to select more than one section.
2. Drag the selected section to the desired position with your pointer.

Delete Header Section

To delete the section in the application, perform the following steps:

1. Navigate to the section node in the outline.
2. Click the **Delete** icon.

   The **Delete Confirmation** pop-up window appears.
3. Click **Delete** to confirm the action.

ℹ Note

This action deletes respective `UI.ReferenceFacet` record from `UI.Facets`.

Note

To remove unreferenced `UI.FieldGroup` annotation, run the cleanup procedure that deletes the unreferenced annotation.

Maintain Header Section Properties

Label

All the **Header** sections regardless the type contain the **Label** property. It can be based on a different annotation properties depending on the section type. For example, the **Label** for the **Form** section is based on the **Label** property of the `UI.ReferenceFacet` record within the `UI.Facets` annotation since the **Label** of the **Data Point** section is based on the **Title** property of the `UI.Chart` annotation. Nevertheless, you can maintain the **Label** for all the **Header** section types in the same way. All the other **Header** section properties depend on the **Header** section type.

To change the section label, perform the following steps:

1. Select the required section and navigate to the properties pane area.
2. Enter a new name in the **Label** text box. This field defines the text to be displayed at a section label.

   As a result, the section is renamed both in the **Page Editor** and in the application preview.

See Label Maintenance for more information.

Note

See Internationalization (i18n) for translation if not yet there.

Sections

The following sections can be created, modified, or deleted in the form and object page:

- Form Section
- Table Section
- Identification Section
- Group Section
- Custom section

Form Section

Form sections are based on the annotation `UI.FieldGroup` and can be added, moved, changed, and deleted.

Add Form Section

To add a Form section, perform the following steps:

1. Click the **Object/Form Entry Page** to open the **Page Editor**.
2. Navigate to the section node in the outline and click the **Add** icon [+].

As a result, a drop-down menu displaying currently supported section types appears.

3. Select **Add Form Section** from the drop-down list.

   A pop-up window **Add Form Section** appears with a field to provide a label for the section to be added.

4. Enter a title to the **Label** field and click **Add**.

   > **Note**
   >
   > See Internationalization (i18n) for translation if not yet there.

   A new section tab appears in the application preview of the form and **Object Page**. Now you can add fields to the newly created form section. For more information, see Adding Filter Fields.

In the `annotation` file, you can see the following changes applied:

- A new `UI.FieldGroup` with the empty `Data` property is added.
- A `UI.ReferenceFacet` record is added to the `UI.Facet` annotation with the following properties:
  - `Target`, as annotation path pointing to the created `UI.FieldGroup`.
  - `Label`, as a string value containing user-entered text.
  - `ID`, as a string value auto-generated based on the label.
- If the `UI.Facet` annotation isn't yet available, it's applied to the entity associated with the **Object Page**.
- If the `UI.Facet` annotation exists on the underlying layer, the annotation on this layer will be overridden in the local file.
- In the case of CAP CDS, a using statement is added to the overridden file.

Move Form Section

The user can change the order of the sections in the application header. By using the drag-and-drop functionality, drag the required section to a different position within the **Header Sections** node:

- When dropped, the records in the `UI.Facets` collection are reordered.
- When SAP Fiori application is rendered, sections are displayed based on the records sequence in the `UI.HeaderFacets` annotation.

Move multiple sections

To move the multiple sections to another position, perform the following steps:

1. Use the Ctrl + Click combination to select more than one section.
2. Drag the selected section to the desired position with your pointer.

Delete Form Section

To delete the section in the application, perform the following steps:

1. Navigate to the section node in the outline.
2. Click the **Delete** icon.

   The **Delete Confirmation** pop-up window appears.

3. Click **Delete** to confirm the action.

ℹ Note

This action deletes respective `UI.ReferenceFacet` record from `UI.Facets`.

ℹ Note

To remove unreferenced `UI.FieldGroup` annotation, run the cleanup procedure that deletes the unreferenced annotation.

Maintain Form Section Properties

Label

To change the section label, perform the following steps:

1. Select the required section and navigate to the properties pane area.
2. Enter a new name in the **Label** text box. This field defines the text to be displayed at a section label.

   As a result, the section is renamed both in the **Page Editor** and in the application preview.

See Label Maintenance for more information.

ℹ Note

See Internationalization (i18n) for translation if not yet there.

Display on Demand

`Display On Demand` switch is displayed in the **Property Panel** for the Form Section and Identification Section section if they are used as a sub section in a Group Section. Switch it on to hide the Form Section or Identification Section under the Show More button by default. You need to press the Show More button to make it visible. Switch it off to always display the section.

ℹ Note

By default this property is switched off. Once the user switches it on, an embedded annotation `UI.PartOfPreview` is added to the respective `UI.ReferenceFacet` record with boolean value `false`. The embedded annotation will be removed, if the property is switched off again. The embedded annotation is also removed, if the section is moved and not contained in a Group Section after the move.

Basic Fields

Adding Basic Fields

To add a new field to an existing section, perform the following steps:

1. Expand the required section and navigate your pointer to the field layer .
2. Click the plus icon + to open the **Add Fields** pop-up window.
3. In the **Add Fields** pop-up window, search for or select one or several field from the drop-down menu.
4. Click **Add** to add new fields to the Form section.

ℹ Note

The following entity properties are excluded from the fields dialog creation process:

- Properties of the type `Edm.Guid`.
- Draft specific properties, such as `IsActiveEntity`, `HasActiveEntity`, and `HasDraftEntity`.
- Draft specific navigation properties such as `SiblingEntity`, `DraftAdministrativeData`.
- The properties already referenced in this or other sections.

Moving Basic Fields

To move a field within a section or to a different section in the application, use one of the following functionalities:

- **Drag-and-drop**

  By using the drag-and-drop functionality, drag the required field and place it in a different position within its section or in a different section once the field is highlighted in green.

- **Arrow buttons**

  By using the **Move up** and **Move down** buttons next to the field name, you can move the field either within its section or to the different section.

Move Fields to Another Section

To move a field to another section, drag and drop to a desired section.

ℹ Note

Fields can be moved to a different section if the `FieldGroup` or `Identification` annotation are applied to the **same entity** as the originating section.

Move Multiple Fields

To move the multiple fields to another position, perform the following steps:

1. Use the Ctrl + Click combination to select more than one field.
2. Drag the selected fields to a different position with your pointer/mouse.

Deleting Basic Fields

To delete the fields perform the following steps:

1. Expand the required section and navigate your pointer to the field layer .
2. Click the Delete icon to open the **Delete Confirmation** pop-up window.
3. Click Delete to confirm the action.

   ### ℹ Note

   During deletion of the field, `UI.DataField` record is removed from the `UI.FieldGroup` annotation. The annotations applied to the entity properties are not deleted.

Maintain Basic Field Properties

Field properties are associated with fields in the Field section and the application with the help of annotations. The following field properties can be edited:

- Label
- Restrictions
- Text
- Text Arrangement
- Display Type
- Criticality

Please see Appendix for more information.

Label

To change the section label, perform the following steps:

1. Select the required section and navigate to the properties pane area.
2. Enter a new name in the **Label** text box. This field defines the text to be displayed at a section label.

   As a result, the section is renamed both in the **Page Editor** and in the application preview.

See Label Maintenance for more information.

ℹ Note

See Internationalization (i18n) for translation if not yet there.

Restrictions

Define whether the field input in create/edit mode is mandatory, optional, or read-only. Use field **Restrictions** to control the state of a property.

The following options are displayed in the restriction value drop-down menu:

Option	Description
None	No annotations are applied. Therefore, this field by default is considered optional.
Optional	This field can be left empty, no obligatory data input is required.
Mandatory	This field value must be provided, cannot be empty.
ReadOnly	This field is displayed as read-only data with no editing allowed.

ℹ Note

When a the Object Page entity is not draft enabled (read-only), Display Type and Restrictions fields are not available in the property panel as the fields are not editable and are only used for displaying the value.

ℹ Note

If restriction value is defined in the lower layer (e.g. in the service), the respective option is displayed with the suffix (base layer) and option **None** is not available. If the backend restriction value cannot be resolved due to unsupported annotation complexity, then the base layer value is displayed as a **Complex**(base layer).

Contact Field

A Fiori application can display contact information as a quick view for a form or Identification Section of an **Object Page** as a `Contact Field`.

Adding Contact Field

To add a contact field to a section, perform the following steps:

1. Click Add Contact Field when choosing + button in Form or Identification node in the **Page Editor** .
2. Select `Contact` via a tree control.
3. Click Add, a new `Communication.Contact` annotation is created.

ℹ Note

Contact fields are based on the `UI.DataFieldForAnnotation` record type contained in the annotations `UI.FieldGroup/UI.Identification` ("base annotation") and reference a `Communication.Contact` annotation in the `Target` property.

Moving Contact Field

A contact field can be moved within a section as any other field. It can be moved between sections unless forbidden by the relative paths of `Communication.Contact` and the base annotations.

Deleting Contact Field

To delete the contact column/field in the application, perform the following steps:

1. Navigate to a field.
2. Click Delete icon.

   The **Delete Confirmation** pop-up window appears.
3. Click Delete to confirm the action.

Table Section

Table section can be added either on the section node or inside the group on the subsection node.

- Add Table Section
- Move Table Section
- Change Table Section Label

- Delete Table Section

Add Table Section

To add a table section, perform the following steps:

1. Click the **Object/Form Entry Page** to open the **Page Editor**.
2. Navigate to the section node in the outline and click the **Add** icon [+].

   As a result, a drop-down menu displaying currently supported section types appears.
3. Select **Add Table Section** from the drop-down list.

   A pop-up window **Add Table Section** appears.
4. Enter a title in the **Label** text box.
5. Then, enter **Value Source Entity** to define the table data source and click **Add**.

As a result, you can see the following changes applied:

- A new `UI.LineItem` with empty collection and a new reference facet with an `annotationPath` pointing to the created `UI.LineItem` is added to the existing `UI.Facets`.
- If not yet available, a new `UI.Facets` annotation is created under the entity associated with that **Object Page**.
- If `UI.Facets` exists on an underlying layer, the annotation in the underlying layer will be overridden.
- For CAP CDS, a using statement is added to the overridden file if not yet there.

Move Table Section

The user can change the order of the sections created in the application. By using the drag-and-drop functionality, drag the required section to a different position within its application:

- When dropped, the records in the `UI.Facets` collection are reordered.
- When SAP Fiori application is rendered, sections are displayed based on the records sequence in the `UI.Facets` annotation.

Move multiple sections

Annotation Library supports mass moving of the sections. To move the multiple sections to another position, perform the following steps:

1. Use the [Ctrl] + [Click] combination to select more than one section.
2. Drag the selected section to a different position with your pointer/mouse.

Delete Table Section

To delete the section in the application, perform the following steps:

1. Navigate to the section layer.
2. Click the **Delete** icon.

   The **Delete Confirmation** pop-up window appears.

3. Click **Delete** to confirm the action.

> **Note**
>
> This action deletes the referenced facet record from `UI.Facets` of the section.

> **Note**
>
> To clean up the orphaned `UI.LineItem` annotation, you need to explicitly run the cleanup procedure that deletes the unreferenced annotation.

Maintain Table Section Properties

Label

To change the section label, perform the following steps:

1. Select the required section and navigate to the properties pane area.
2. Enter a new name in the **Label** text box. This field defines the text to be displayed at a section label.

   As a result, the section is renamed both in the **Page Editor** and in the application preview.

See Label Maintenance for more information.

> **Note**
>
> See Internationalization (i18n) for translation if not yet there.

Related Information

Table Actions
Table Columns

Identification Section

Add Identification Section

An Identification Section can be added if `UI.Identification` annotation isn't yet defined for the page entity or not referenced in `UI.Facets` annotation.

> **Note**
>
> More than one identification section can't be added for a page. Identification sections cannot be added as subsections. Actions cannot be added to the identification section. Header or Footer actions can be added instead.

To add an Identification section, perform the following steps:

1. Click the **Object/Form Entry Page** to open the **Page Editor**.
2. Navigate to the section node in the outline and click the **Add** icon [+].

   As a result, a drop-down menu displaying currently supported section type appears.

3. Select **Add Identification Section** from the drop-down list.

   A pop-up window **Add Identification Section** appears with a field to provide a label for the section to be added.

4. Enter a title to the **Label** field and click **Add**.

   A new section tab appears in the application preview of the form and object page.

> **Note**
>
> As a result a `UI.Identification` annotation with no qualifier is generated and referenced in the `UI.Facets` annotation. If `UI.Identification` already exists, a new one isn't generated, but the existing one is referenced in `UI.Facets`.

You can prepare the section label for translation.

- For more information, see Internationalization (i18n).

Move Identification Section

The user can change the order of the sections created in the application. By using the drag-and-drop functionality, drag the required section to a different position within its application:

- When dropped, the records in the `UI.Facets` collection are reordered.
- When SAP Fiori application is rendered, sections are displayed based on the records sequence in the `UI.Facets` annotation.

Move multiple sections

To move the multiple sections to another position, perform the following steps:

1. Use the Ctrl + Click combination to select more than one section.
2. Drag the selected section to a different position with your pointer/mouse.

Delete Identification Section

To delete the section in the application, perform the following steps:

1. Navigate to the section node in the outline.
2. Click the **Delete** icon.

   The **Delete Confirmation** pop-up window appears.

3. Click **Delete** to confirm the action.

> **Note**
>
> This action deletes respective `UI.ReferenceFacet` record from `UI.Facets`.

Maintain Identification Section Properties

Label

To change the section label, perform the following steps:

1. Select the required section and navigate to the properties pane area.
2. Enter a new name in the **Label** text box. This field defines the text to be displayed at a section label.

   As a result, the section is renamed both in the **Page Editor** and in the application preview.

See Label Maintenance for more information.

> **Note**
>
> See Internationalization (i18n) for translation if not yet there.

Display on Demand

See Display on Demand.

Chart Section

Chart section can be added either on the section node or inside the group on the subsection node.

Add Chart Section

To add a chart section, perform the following steps:

1. Click the **Object/Form Entry Page** to open the **Page Editor**.
2. Navigate to the section node in the outline and click the **Add** icon [+].

   A drop-down menu displaying currently supported section types is displayed.
3. Select **Add Chart Section** from the drop-down list.

   A pop-up window with following attributes:

   - **Label**
   - **Entity**
   - **Type**
   - **Dimension**
   - **Measure**

   A measure can be specified by selecting one of the following:

   - Use existing measure
   - Create new measure

   If you choose to use existing measure, select one of the available measures defined with custom or transformation aggregations in the **Name** field.

   If you choose to create new measure, choose the aggregatable property and one of the supported aggregation methods . This allows you to create a new **dynamic** measure and use it in the chart.

   > **Note**
   >
   > The technical name and the label are generated automatically . You can then adjust the generated label in the **Property Panel**.

4. Click **Add**

As a result, you can see the following changes applied:

- A new `UI.Chart` and a new `UI.ReferenceFacet` with an `annotationPath` pointing to the created `UI.Chart` is added to the existing `UI.Facets`.
- If you chose to create a new measure, `@Analytics.AggregatedProperty` is applied to the selected aggregatable property with your chosen aggregation method.
- If not yet available, a new `UI.Facets` annotation is created under the entity associated with that **Object Page**.
- If `UI.Facets` exists on an underlying layer, the annotation in the underlying layer is overridden.
- For CAP CDS, a using statement is added to the overridden file if not yet there.

Delete Chart Section

You can delete section that is not required to be displayed in the UI.

- In cases of deletion, the generated `UI.ReferenceFacet` is deleted from the `UI.Facets` collection.
- The `UI.Chart` annotation is cleaned up during the cleanup procedure.

Maintain Chart Section Properties

You can update the properties of the chart and define optional properties in the **Property Panel**. For the information on changing the section label see Change Form Section Label

For the information on maintaining other chart properties see Maintain Analytical Chart Properties.

Group Section

A Group section groups multiple sections together. It can contain a group of sections of any type except custom sections. The group section can't contain fields/columns or actions. With several nested group sections, you can build a nested tree of sections.

ⓘ Note

Fiori elements may not render group sections beyond the certain level and have rendering limitations on displaying Table sections as peer to the form sections. Please check the current documentation for Defining and Adapting Sections.

Add Group Section or Subsections

To add a Group section or subsections, perform the following steps:

1. Click the Object/Form Entry Page to open the **Page Editor**.
2. Navigate to the section layer and click the **Add** icon +.

   As a result, a drop-down menu displaying currently supported section type appears.
3. Select **Add Group Section** from the drop-down list.

   A new pop-up window **Add Group Section** appears with a field to provide a label for the section to be added.
4. Enter a title in the **Label** text box and click Add.

ⓘ Note

Internally, a new entry is added to the annotation `UI.Facets`. This entry is of the type `UI.CollectionFacet` with corresponding property Label and property Facets being set to an empty array. If `UI.Facets` doesn't exist yet or is not present in the changeable annotation file, managing process is the same as when adding a new Form section.

A new Group tab appears in the application preview.

Add Subsections

1. Navigate to the Subsections node and click the **Add** icon [+].
2. Select the required section from the list, such as Form Section.

   A new pop-up window **Add Form Section** appears with a field to provide a label for the section to be added.
3. Enter a title in the **Label** text box and click [Add].

ⓘ Note

In the new Form section, you can perform the same actions as in the classic Form Section , such as adding, editing, moving, and deleting fields.

Label properties can be prepared for translation. For more information, see Internationalization (i18n). In addition, see Edit in Source Code feature to navigate to code fragments in the annotation file.

Move Sections

Sections in the Group section can be moved as follows:

- Within a Group section.
- Between different Group sections.
- To the top level and back.

By using the drag-and-drop functionality, drag the required section to a different position.

For more information, see Move Basic Fields.

Delete Group Section or Subsection

To delete the section in the application, perform the following steps:

1. Navigate to the section layer.
2. Click the [Delete] icon.

   The **Delete Confirmation** pop-up window appears.
3. Click [Delete] to confirm the action.

ⓘ Note

During deletion of the group section, the respective `UI.CollectionFacet` record is deleted from the `UI.Facets` annotation along with all its content.

ⓘ Note

To clean up the unreferenced annotations for the deleted section content, you need to run the cleanup procedure that deletes the unreferenced annotation.

Maintain Group Section Properties

Label

To change the section label, perform the following steps:

1. Select the required section and navigate to the properties pane area.
2. Enter a new name in the **Label** text box. This field defines the text to be displayed at a section label.

   As a result, the section is renamed both in the **Page Editor** and in the application preview.

See Label Maintenance for more information.

Note

See Internationalization (i18n) for translation if not yet there.

Appendix

Criticality Representation

When **Criticality** property is defined for the basic table column or section field, the **Criticality Representation** property appears in the properties pane right after it. If you want to use the default representation defined by Fiori elements, you can leave this field set to **None**. Otherwise, you can explicitly define whether to indicate the criticality by an icon in addition to the semantic coloring. For this, choose one of the following options:

- **With Icon** – displays the icon in addition to the semantic coloring, independent on the default representation defined by Fiori elements.
- **Without Icon**

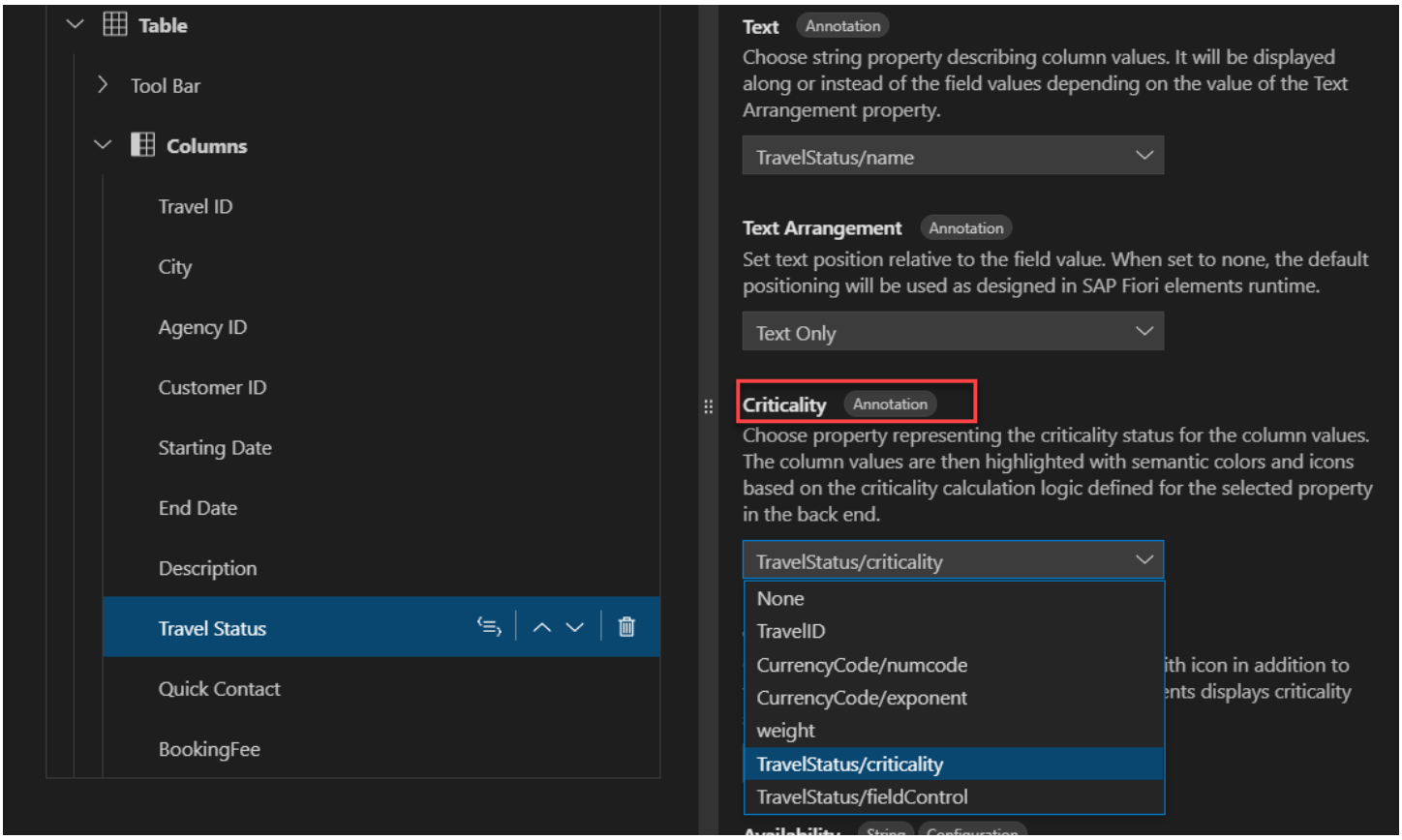
As a result, **Criticality Representation** property is added to the `UI.DataField` record and the respective column or field values are shown with or without the icon, independent on the default representation defined by Fiori elements presenting the table column or section field.

Criticality

You can display the values of the section fields or basic table columns with semantic coloring and optionally with criticality icons. For example, you can choose to display the **Travel Status** value in red if the trip is cancelled and green if it is confirmed.

As a prerequisite, your service should contain the property representing the status criticality information. If this prerequisite is fulfilled, do the following:

- In outline, choose the table column or field you want to show with the semantic information.
- In the Properties pane, choose the property representing the status criticality information in the **Criticality** field.



As a result, **Criticality** property is added to the `UI.DataField` record and the respective column or field values are shown in semantic colors. In addition, the criticality icon may appear, that depends on the default behavior of Fiori elements template. You may override this default by explicitly defining the criticality representation.

Description

In addition to the section label, some header section types, such as progress or micro chart, let you set the description of the content. To define the description:

1. Open the **Page Editor**, choose the Header Section.
2. In the Properties Pane, enter the text providing additional information for your content in the Description property.

ⓘ Note

Similar to section label, description text can be prepared for translation. For more information see Internationalization (i18n).

Text

Fields and basic table columns representing IDs or codes often need to be displayed along with the descriptive text which conveys the meaning in a human-readable way. For example, status codes - `0`, `A`, `C`- could be meaningless for the user and should be accompanied or even replaced by the descriptive text, such as Open, Accepted, Cancelled.

To add such descriptive texts, select the property representing the descriptive text in the **Text** property. Then, the `Common.Text` annotation will be applied to the property representing field/column value.

Text Arrangement

When Text is defined, the `Text Arrangement` property appears in the properties pane right after it. If you want to use the default arrangement of Fiori elements, you can leave this field set to None. Otherwise, you can define how this text is displayed to the field or column value.

ℹ Note

The option None is not available if `UI.TextArrangement` or `Common.TextArrangement` is already defined on a lower layer, such as in the service.

Also, you can set the text arrangement explicitly, as follows:

- To display both, the field/column value and text, select the Text First or Text Last values.
- To substitute the field/column value with the text, select Text Only.
- To conceal the property defined as a text in the value help, select ID Only.

As a result, the `UI.TextArrangement` annotation is applied to the `Common.Text` annotation defined for the text property.

Text and Text Arrangement for Fields Configured with ValueHelp

If the field display type is configured the same way as `Value Help`, you may want to configure the field value the same way as the `Value Help` value.

To do so, select the same property as Text for the field and Value Description Property for `Value Help`. Also, choose the same options for the text arrangement.

- **Example 1**:

  The selected value in the filter field "Type" is consistent with the `Value Help` list values.

  This is assured by the same values selected in Text/Value Description Property and Text Arrangement for the filter field and its value help. The `typedescription` property is used as Text/Value Description Property and Text Only as Text Arrangement.

  Filter Field properties:

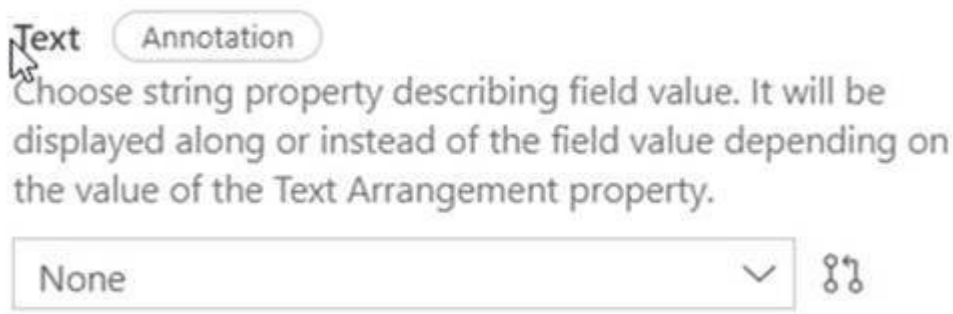
`Value Help` properties:

- **Example 2**:

  Selected value in the filter field is not consistent with the value help list values.

The Text/Value Description Property and Text Arrangement are configured differently on the field and value help. The `typedescription` property is used as Value Description Property and `Text Only` as Text Arrangement for the value help, while Text and Text Arrangement are not defined on the filter field itself as set to None.

Filter Field properties:



Value Help Properties:

The button is provided by the Page Editor to simplify the text/text arrangement synchronization. It appears next to the Text field once the Value Description Property is defined in the `Value Help`. When you click it, the value defined in the `Value Help` is set as Text for the field.

Similarly, it appears next to the Value Description Property field in the `Value Help`, once Text is defined for the field. When you click it, the value defined for the field Text is set for the Value Description Property .

ℹ Note

The button doesn't appear if both properties are set in the same way.

If the values of Value Description Property and Text field are synchronized, Text Arrangement values are checked. If they don't match, the buttons appear next to the Text Arrangement allowing you to synchronize one value with another.

As a result, the values in the `Common.Text` annotations applied to the field value and source value of the value help point to the same property and `UI.TextArrangement` have the same enum value.

- Text and Text Arrangement on the filed value:

```
annotate service.CapexType with {
   type @Common.Text : {
       $value : typedescription,
       ![@UI.TextArrangement] : #TextOnly,
   }
};
```

- Text and Text Arrangement on the source value of the value help:

```
annotate service.Capex with {
    type @Common.Text : {
        $value : type.typedescription,
        ![@UI.TextArrangement] : #TextOnly,
    }
};
```

Label Maintenance

The `Form/Table/Identification/Chart` section labels are translatable and readable elements which are rendered in the SAP Fiori apps. When the user creates a supported section, the label inputted by the user is assigned to the `Label` property of the `ReferenceFacet` record in the `Facet` annotation. During deletion of the section, the `ReferenceFacet` record is deleted which eventually deletes the `Label` in it.

Note

All the labels translatable and can maintained through project i18n files. See Internationalization (i18n).

Fields

- The `Fields` labels can be maintained with annotations, such as `Common.Label` and `@title`.
- The `Fields` sublabel can be maintained with type information of the entity property such as `name: String(50)`.
- The navigation to the source code leads to the respective `DataField` record.
- In case, these annotations are not present for the entity property, the `Label` property of the `DataField` is generated with the same value as the `Value` property.
- The application does not generate the label annotations directly on the properties.
- If you attempt to change the value of the Label provided by annotation, the annotation value is not updated. Rather the `Label` property of `DataField` is generated or updated.
- During deletion of the field, annotations mentioned above are not deleted, only labels which are directly maintained in record are deleted as record is fully completely removed. See Internationalization (i18n) for more information.

Columns

The `Column` labels can be maintained with annotations, such as `Common.Label` and `@title`. The `Basic Column`sublabel can be maintained with type information of the entity property such as `name: String(50)`. The navigation to the source code leads to the respective `DataField`record. The other column types such as `Rating Column`, `Progress Column` sublabel can be maintained with column type information e.g. `Type: Rating` or `Type: Progress`. In case the annotations are not maintained, the label would be empty and the user can add label through the property panel and the Label property for DataField is generated. During deletion of column, the label maintained in record gets deleted as the record is completely removed.

Actions

`Actions` has also editable labels.

- During creation of action, we use the the last segment of actions to create the label. For example, `Trippin.Container/GetNearestAirport` the `Label` will have `GetNearestAirport` as the value assigned to it.
- This Label would be eventually rendered as the Button Label in Fiori application. During deletion the the entire `DataFieldForAction` record is deleted, thus deleting the `Label` along with it.
- Label based annotations are not removed during the cleanup procedure Project Cleanup.

Measures and Currencies

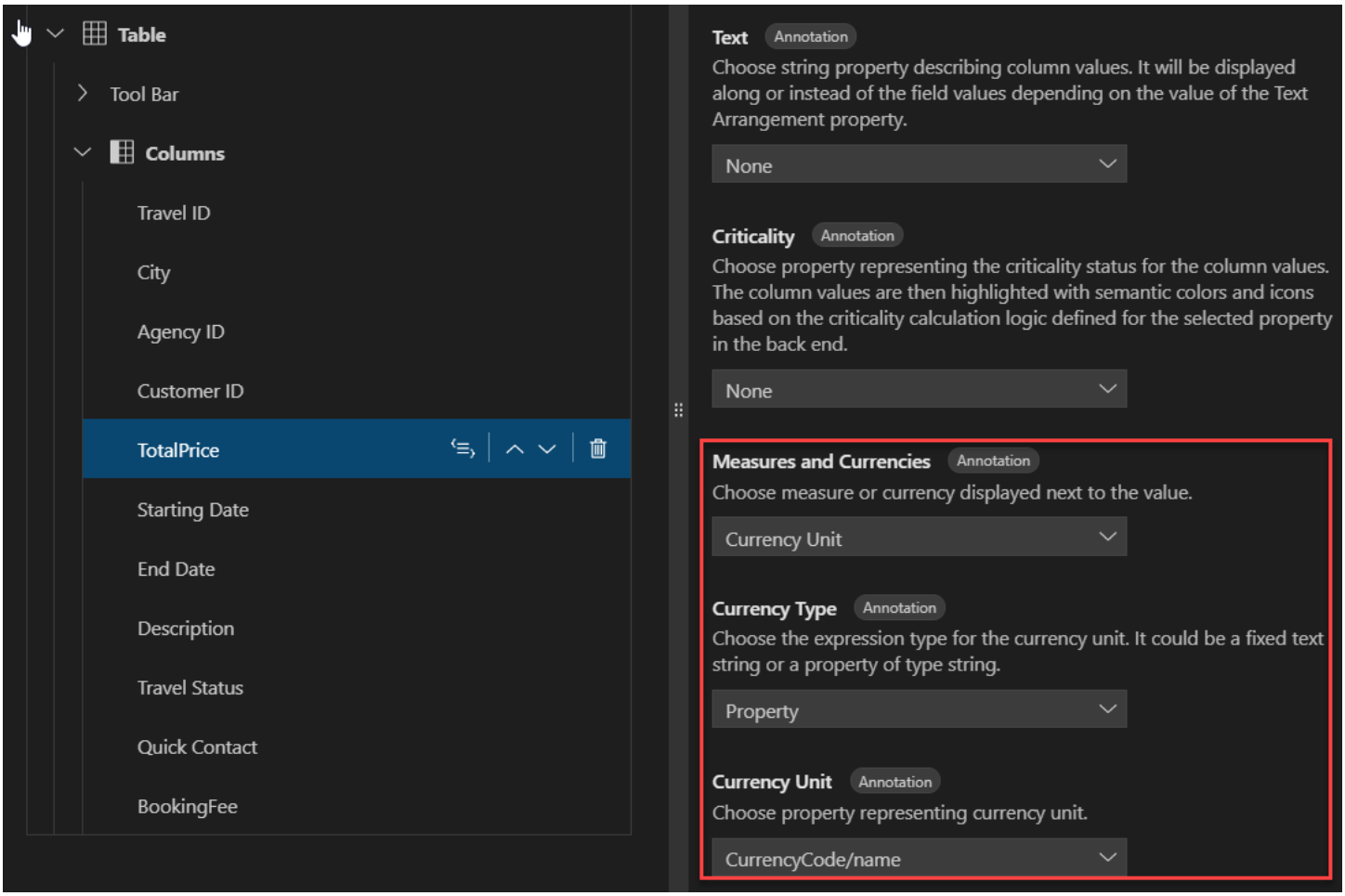
You can display the numeric values of the section fields or basic table columns together with measures or currencies represented by these values. For example, you can display prices along with the currencies and product dimensions, such as width or weight, with the measure unit. For this, As a prerequisite, do the following:

- In outline, choose the table column or field you want to show with the semantic information.
- In the Properties pane, choose one of the following options in the **Measures and Currencies** field:
    - **Currency Unit**
    - **Measure Unit**
- In the pop-up, define how the unit should be represented by choosing one of the following options and choose `Apply`:

- **Path** – if you want to define the unit as the property of associated entity. In this case, choose the property representing measure or currency units.
- **String**– if you want to define the unit as plain text, such as %. the property of associated entity. In this case, enter the text for the unit to be displayed along with the value

As a result, `Measures.ISOCurrency` annotation is applied to the field or column value referencing the property or string you chose and the respective column or field values are shown with the respective currency or measure unit.

You can change the selected measure or currency in the properties pane at any time by choosing the values in the newly appeared Type and Unit fields. You can prevent displaying the unit with your column/field value by setting the Measures and Currencies value to None.



Tooltip

You can set tooltips for some table columns and header section types, in particular:

- Progress Columns in List Report and Object Page tables.
- Rating Columns in List Report andObject Page tables.
- Progress Header Sections in Object Page.
- Rating Header Sections in Object Page.
- Data Point Header sections in Object Page.

This tooltip can be set as a fixed text or as a dynamic text coming from the service property.

To set the tooltip:

1. Open the **Page Editor**, choose the Header Section or Column of type Supporting tooltips.
2. In the Properties Pane, choose select the desired source of the tooltip.
    - **String**: lets you enter the fixed translatable text.
    - **Property**: provides a list of string service properties to choose from.
3. Enter the tooltip text or choose the desired property based on the option you selected.

ⓘ Note

Tooltip defined as text string can be prepared for translation. For more information see Internationalization (i18n).

ValueHelp

You can configure the value help for the section fields, table columns, and filter fields unless they're represented by properties of type Boolean or defined as read-only (directly or via the parent entity). To do so, you need to set the **Display Type** property to **Value help**.

To enable the value help, your service should contain the entity set representing the list of suitable values. For example, if you want to define the value help for the CapexType field, your service should have an entity set, such as CapexType containing at least one property representing available CAPEX categories.

```
entity CapexType : managed {
    key type            : String;
        typedescription : String;
}
```

When the **Value help** option is selected, the dialog window **Define Value Help Properties** appears where you provide the data source parameters for possible values:

- **Value Source Entity**. Entity set representing the list of field values.

  If you work in the CAP project and the field value is defined as an association, the associated entity is suggested automatically. For example, if you configure the Value Help for the Type field defined as an association to the CapexType, Capex type will be added automatically as a value here.

  ```
  entity CapexBase : managed {
          type                : Association to CapexType;
  }
  ```

- **Value Source Property**. Property to be used as an input field.

  If you work in the CAP project and the field value is defined as an association, the first key property of the associated entity is automatically suggested.

- **Value Description Property**. Property to be used for displaying the additional text along with or instead of the input value.

  This property is usually defined if an input value **Value Source Property** represents a code or ID, and serves for explaining the meaning of that code/ID. For example, if **Value Source Property** is set to the type field representing some CapexType

code, we recommend that you choose in Value Description Property representing the human-readable description of the Capex type.

> **Note**
>
> Value Description Property and Text Arrangement are similar to Text and Text Arrangement properties of Filter fields, Form section fields, and basic table columns. They result in the same annotations and are applied to the property selected as Value Source Property. If you expect the Text and Text Arrangement defined for the field to be the same as in the value help, click the Take Over button to apply the respective values.

- **Text Arrangement**. Defines how the Value Description Property is displayed with regards to the Value Source Property.

  You can display them together by selecting the Text First or Text Last values or substitute the code/ID represented in Value Source Property with the descriptive value by selecting Text Only.

  > **Note**
  >
  > If you select ID only, Value Description Property isn't displayed in the value help.

- Display as Dropdown checkbox. Defines if the field is displayed as a combo-box or a standard value help dialog.

  > **Note**
  >
  > Check SAP Fiori guidelines to decide which option to use.

- Result List

  You can let your user differentiate the available options by configuring the Result List table. For example, it could contain instructions on when the specific type applies and the processing rules, if any.

  To add table columns to the Result List table, choose the Add Column button and select a property for it.

  If needed, you can set the dependent filtering. For this, you choose the dependency direction in the Dependency column, and the respective local property in the Local Value column:

  - None: the selected property is represented in `Common.ValueList` annotation as `ValueListParameterDisplayOnly`. At runtime, the selected value doesn't affect other fields or columns based on the same property. The Local Value column isn't applicable in this case.
  - In: the selected property is represented in `Common.ValueList` annotation as `ValueListParameterIn`. At runtime, the selected value filters the available options in other fields or columns based on the same property. When this dependency is selected, you must choose the corresponding property from the main entity in the Local Value column.
  - Out: the selected property is represented in `Common.ValueList` annotation as `ValueListParameterOut`. The selected value will be automatically set in other fields or columns in runtime based on the same property. When this dependency is selected, you must choose the corresponding property from the main entity in the Local Value column.
  - InOut: the selected property is represented in `Common.ValueList` annotation as `ValueListParameterInOut`. At runtime, the selected value gets both automatically set and filters the available options in other fields or columns based on the same property. When this dependency is selected, you must choose the corresponding property from the main entity in the Local Value column.

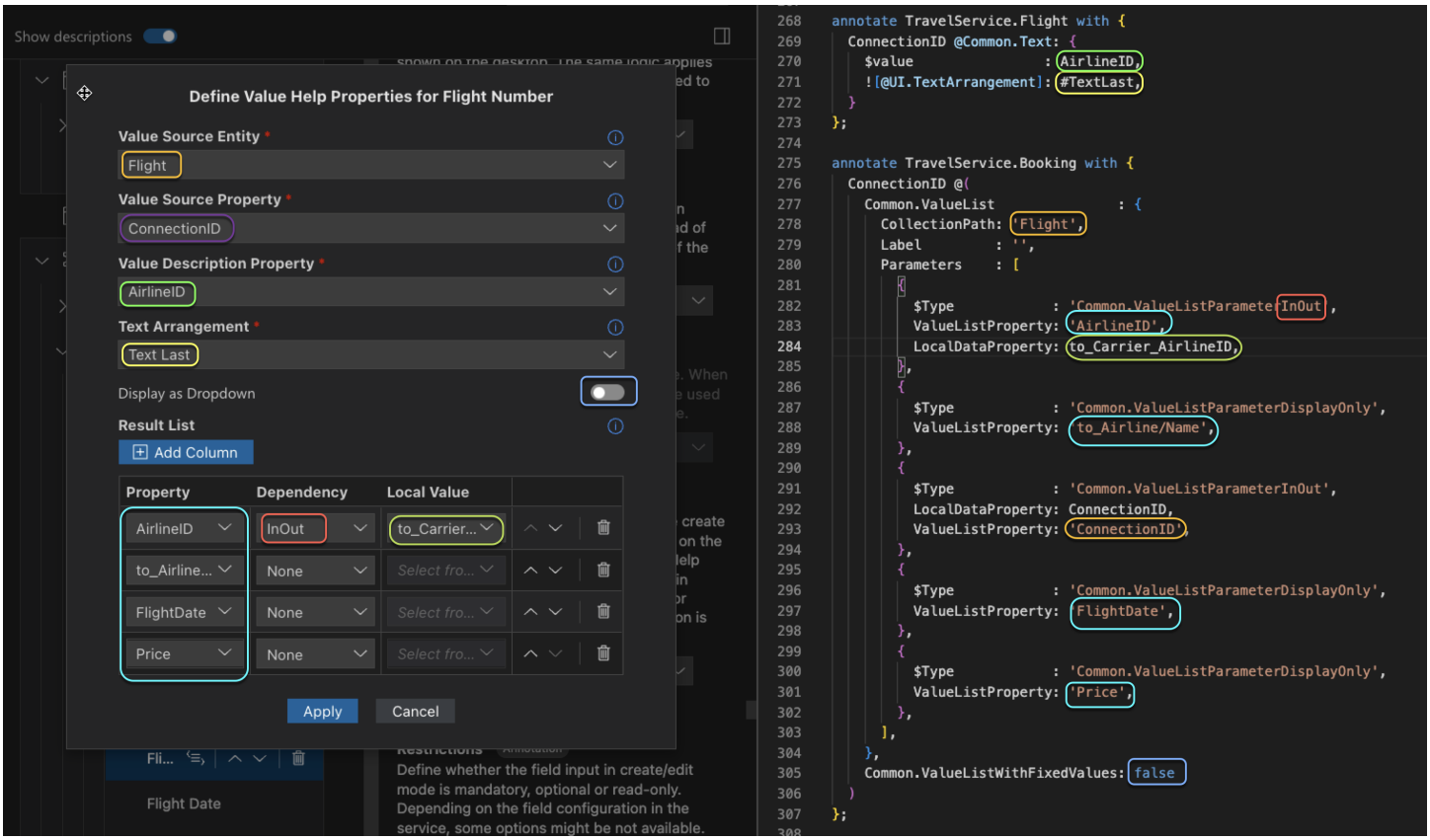
Select the required value and click Apply.

When the configuration is done, the following corresponding annotations are generated/updated on the application layer:

- `UI.MultiLineText`

- `Common.ValueList`
- `Common.ValueListWithFixedValues`
- `Common.Text`

These annotations map to the Value Help properties as follows:



ℹ Note

To edit previously selected properties, click Edit properties for Value Help.

Use Feature Guides

With Guided Development, you can access **how-to** guides and tutorials that explain how to implement certain functionality in an SAP Fiori elements application. You can read through the steps required to implement a feature and then use the guided development approach to make the required changes in your project.

Launching Guided Development

Guided Development can be launched in several ways.

Using Command Palette

- Open Command Palette (CMD / CTRL + Shift + P).
- Start typing ***Guided Development***.
- Select Fiori: Open Guided Development or Fiori: Open Guided Development to the Side.
- Select SAP Fiori elements project from your workspace.

The Fiori: Open Guided Development option opens Guided Development in a new tab. The Fiori: Open Guided Development to the Side option opens Guided Development to the side of the current file in another column.

Using folder context menu

If you already have a SAP Fiori elements project in your current workspace, you can right-click its folder and select SAP Fiori tools - Open Guided Development. Then, Guided Development opens to the side of the current file in another column.

> **Note**
>
> If you do not have any SAP Fiori elements project in your workspace, you can still open Guided Development by using Command Palette. It is possible to check the available guides descriptions and code samples, while interactive features are disabled in this case.

Working with projects

Guided Development can only work with one project at a time. The name of this project is displayed on the tab header next to **Guided Development**, and this project provides all project-specific data used in the guides.

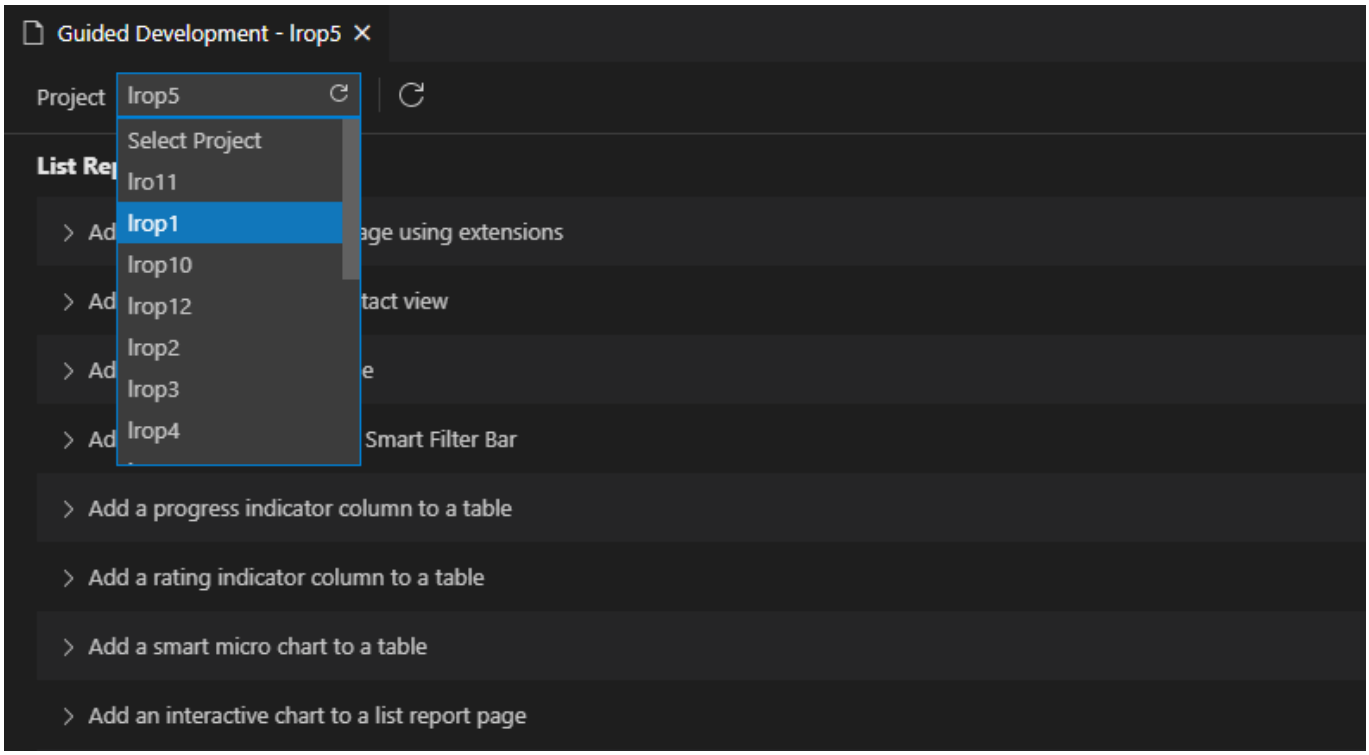
The project-specific data contains the following components:

- The list of entities in the Entity list.
- The list of data sources in the Model list.
- The list of pages in the Page list.
- The annotation terms that are defined in the service across all the guides.

> **Note**
>
> In Guided Development, you can add annotations from services other than the mainService.

To select or change a project.

1. Click Select Project on the left side of the toolbar.
2. Select a project from the Project list.



In some cases, we recommend that you refresh **Guided Development**.

When	How
When a new project is added to the workspace.	Click the “Refresh” icon inside the Project list.
When something in the current project is changed outside Guided Development, such as a new page added, or an underlying service is updated.	Click the “Refresh” icon on the toolbar next to the Project list.

Accessibility

Guided Development can be navigated with either a mouse or a keyboard. Keyboard navigation provides a streamlined experience, allowing users to find and use guides without needing to use their mouse. Use the arrow keys to navigate within sections, the *Tab* key to navigate to new sections and controls, *Shift + Tab* to navigate back to sections and controls, and *Enter* to make selections.

Guided Development supports the use of high contrast themes.

Search for a Guide

On the main screen of the Guided Development extension, you can see a list of available guides. To see a brief description of a feature to implement and its sample preview, expand the respective entry. To get to the guide itself, click **Start Guide** under the description.

Each guide has **categories**, **tags**, and other attributes. A **category** reflects an OData version, template, and annotations type the guide is relevant for. A **Tag** is a kind of metadata assigned to a piece of information in the guides. Both **categories** and **tags** make searching for a guide easier.

Group by

The **Group by** list changes how the guides are grouped in the list based on their categories.

- **Page Type** refers to the page or template. Some features can be applied to different page types, and such guides are displayed under all categories that they're relevant for.

  When a guide available for multiple page types is selected, only the guide relevant to the page type of the current project is highlighted.

- **OData version** refers to the OData service version for which the guide steps are relevant. If a selected guide isn't currently available for a given OData version, a warning message appears.

  Some features are available for both V2 and V4 OData versions and the guides for these features display under both categories. The instructions in such guides could be different for `OData V4` and `OData V2`. In such cases, you can find a guide relevant to your OData version by looking for the right category.

- **Application Artifacts** refers to the type of change the guide describes. The options include manifest change, flex change, and annotations.

Annotations-specific guides are further grouped into three categories: XML Annotations, CDS Annotations, and ABAP CDS Annotations.

For features that can be implemented via different annotation types (depending on what is applicable your project); guides may display under multiple categories. For example, a guide may appear under both the XML Annotations and CDS Annotations categories if variants for the two different annotation types exist. The parameters and code snippets will differ depending on which guide variant is opened.

View

By default, only the guides that are relevant to the current project are displayed. This ensures that code inserted into your project is appropriate for the project's template, OData version, and UI5 version.

Using the **View** filter, you can see all the guides available in Guided Development.

To start working with the **View** filter, perform the following steps:

1. Select a project from the **Project** dropdown list in the upper-left corner.
2. In the **View** dropdown list, pick from the following options:
   - **All Guides**. This option displays the list of all the guides in Guided Development.

     ℹ Note

     When **All Guides** are selected, the updated list of guides appears with the information icon and message similar to the following:

     `Not all guides are applicable to [project name]. Click here to see all project guides.`

     Click the provided link to switch to the **Project Guides** list.

     The guides that aren't relevant to the current project are indicated by a warning symbol next to the guide title. After you open the guide, the following warning message can be viewed:

     `This guide isn't compatible with the current project and will not work as intended.`

   - **Project Guides**. This option displays only those guides that apply to the current project.
3. Click the guide applicable to your project and follow its steps to modify your application.

Filter

The Filter  icon next to the View list allows you to select one or more predefined tags assigned to the guides. The guide list is then narrowed down to show only those guides that have all selected tags.

Search guides

The Search guides field allows you to enter free text to search for a specific term in all the guides content, including titles, descriptions, steps, and code snippets. For example, if you look for all the guides that are relevant to the `UI.LineItem` annotation, you can type in the search term and narrow it down to the list of guides using this annotation term. The search works on ***as you type*** principle, so each successive character entered filters the list further.

Currently Available Guides

Below is the list of guides currently available in SAP Fiori tools Guided Development. An X in the column indicates that the annotation type, OData version, or file change is supported.

Guide Variants Available	XML	CAP CDS	ABAP CDS	ODataV2	OData V4	TypeScript	Manifest Change	Page Configuration Change	Extension Point	Flexible Programming Model Building Block
Add a chart building block		X			X					X
Add a custom action to a page using extensions				X	X	X				
Add a custom card to an overview page									X	
Add a custom filter to the filter bar									X	
Add a custom section to an object page using extensions				X	X					
Add a field group as a section to a page	X	X	X							
Add a filter bar building block										X

Guide Variants Available	XML	CAP CDS	ABAP CDS	ODataV2	OData V4	TypeScript	Manifest Change	Page Configuration Change	Extension Point	Flexible Programming Model Building Block
Add a header facet using data points	X	X	X							
Add a link list card to an overview page	X									
Add a list card to an overview page	X									
Add a new column as a contact view	X		X							
Add a new section to a page	X	X	X							
Add a new visual filter	X	X								
Add a progress indicator column to a table	X	X	X							
Add a rating indicator column to a table	X	X	X							
Add a smart chart facet to an object page	X						X			
Add a smart micro chart to a table	X	X	X							
Add a stack card to an overview page	X									
Add a static link list card to an overview page								X		

Guide Variants Available	XML	CAP CDS	ABAP CDS	ODataV2	OData V4	TypeScript	Manifest Change	Page Configuration Change	Extension Point	Flexible Programming Model Building Block
Add a table card to an overview page	X									
Add an action button	X									
Add an analytical card to an overview page	X									
Add and edit filter fields	X	X	X							
Add and edit table columns	X	X	X							
Add custom columns to the table using extensions									X	
Add interactive chart	X	X								
Add multiple fields to a column in responsive tables	X									
Add semantic colors to visual filters	X									
Add semantic highlights to line items in tables based on their criticality	X	X								
Add status colors and icons for a column	X	X	X							

Guide Variants Available	XML	CAP CDS	ABAP CDS	ODataV2	OData V4	TypeScript	Manifest Change	Page Configuration Change	Extension Point	Flexible Programming Model Building Block
Configure flexible column layout							X			
Configure inbound navigation								X		
Configure multiple selection in tables				X	X		X			
Configure multiple views	X	X								
Configure mass edit via dialog								X		
Configure object page header	X	X	X							
Configure outbound navigation	X									
Configure selection of all rows in a table							X			
Configure side effects	X	X								
Configure spreadsheet export								X		
Configure variant management				X	X					
Create annotations for Key Performance Indicator (KPI) tags	X									
Define a filter facet	X									

Guide Variants Available	XML	CAP CDS	ABAP CDS	ODataV2	OData V4	TypeScript	Manifest Change	Page Configuration Change	Extension Point	Flexible Programming Model Building Block
Enable a Show Related Apps button								X		
Enable condensed table layout							X			
Enable data label in analytical charts								X		
Enable data label in smart charts and KPI cards								X		
Enable draft toggle buttons								X		
Enable inline creation of table entries								X		
Enable object creation in a table via dialog							X			
Enable semantic date range on smart filter bar								X		
Enable table to auto load data				X	X					
Extend forms in sections									X	
Extend object page headers using extensions									X	
Reorder columns in a table	X									

Guide Variants Available	XML	CAP CDS	ABAP CDS	ODataV2	OData V4	TypeScript	Manifest Change	Page Configuration Change	Extension Point	Flexible Programming Model Building Block
Set default filter values	X									
Set selection limit for tables								X		
Set the table type of tables				X	X					
Specify layout for the card container								X		
Specify refresh interval for cards								X		

Develop with a Guide

Once the required guide is selected, click **Start Guide** under the description section to start the development process. By default, the guide is opened in a **side-by-side** mode. With this flexible mode, you can see both the guide window and the list of guides window side-by-side in split view.

- You can adjust window by dragging the vertical line between the windows.
- If you want to exit split view, navigate the pointer to the upper-right corner of the guide window and click the full-screen button.

  Note

  If you reduce the size of the application window, the list of guides will be hidden automatically.

- To switch to a normal view:
  - In **Command Palette**, enter **Preferences: Open User Settings**.
  - Click **SAP Fiori tools - Guided Development**.
  - Select **Normal** from the **Select Guide List View** drop-down list.

A guide walks you through details of a specific feature/task that needs to be implemented in the application. It provides a short description of the feature, screenshots, and a reference to additional documentation containing more information on the topic. Links to additional documentation can be found under the Information icon in the top-right corner of the Guide details screen.

If a guide involves creating annotations, it also includes a list of Annotation Terms used to implement the functionality. Navigate your pointer over “Annotation Term” to display a tooltip with more information about it.

To implement a feature, you need to follow all the steps provided in the selected guide.

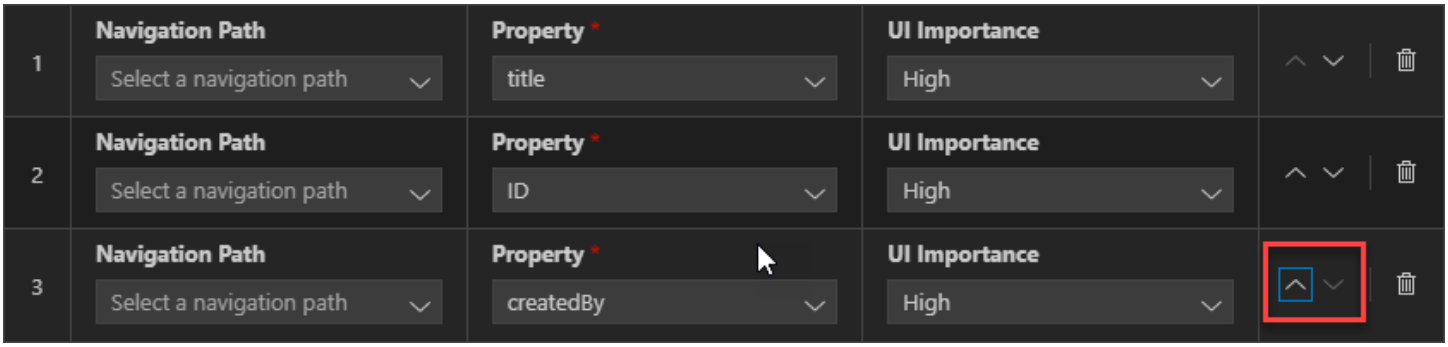
A step includes:

- Details of the file that would need to change.
- A brief description of the change required.
- Code snippet with sample code to implement the step.
- Parameters to customize code snippet.

If required, you can select multiple values from a drop-down list. After you entered all the values, you can either apply the code snippet or copy it based on the instructions in the step. If an error is found with one or more of your parameter selections, Guided Development will scroll up to the parameter that needs correcting. Select another parameter to reactivate the Insert Snippet button.

Some guides may feature a table of parameters related to screen elements in your application. For example: the parameters that make up the columns in a table or the filter (selection) fields available in your project. The different rows represent the order of the elements in the application. New elements can be added using the Add button. For example: Add column or Add selection field. This button will be found above the table, with an additional button present below the table if it has more than 5 rows. This is intended to reduce your need to scroll in the event you have a larger table. In some cases it may also be possible to select from different annotations, for example: Add Data Field or Add Data Field for Annotation, via the dropdown next to the button. The parameters displayed in the new row will update based on your selection.

You can rearrange existing screen elements, as well as the ones added by the guide, by reordering the rows in the table. This can be done by clicking anywhere in the row and dragging it to the desired position. You can also rearrange elements using keyboard navigation. Once your focus is within the collection of parameters, press Tab until your focus is on the parameter rearrangement arrows. The left and right arrow keys are used to switch between the up or down arrows. Press Enter to move the row in the desired direction. Regardless of the method chosen to rearrange the collection of parameters, a loading indicator will indicate that the code snippet is being updated.



You can view the guide in different modes:

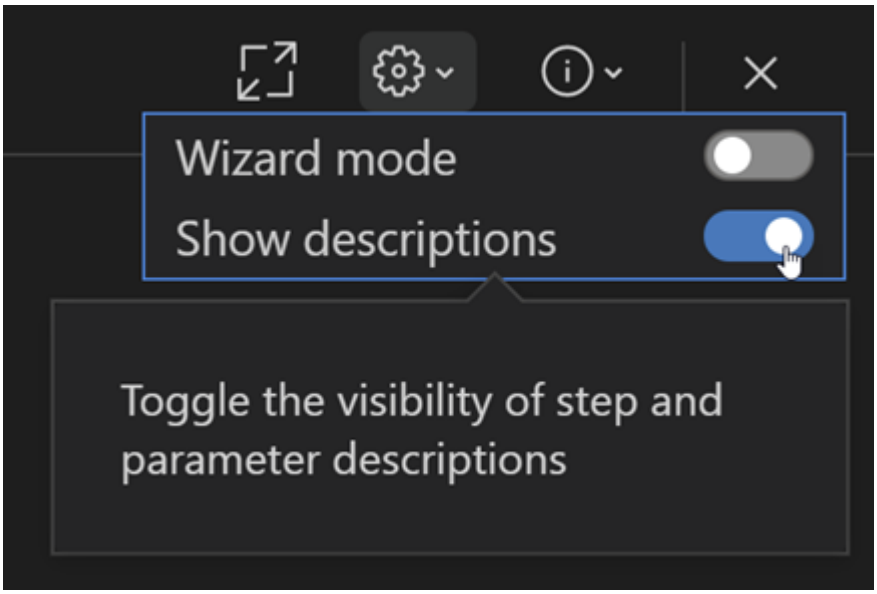
- **Wizard mode (default)**
  - Displays each step in a separate screen.
  - Click the Next button in the bottom left corner to navigate to the next step and the Back button to return the previous step (or guide description if there isn't a previous step).

- You can also click the numbered step tiles at the top to get to any step directly.

- **Full view mode**
  - Displays all steps in one scrollable screen.
  - You can also click the numbered step tiles at the top to get to any step directly.
- **Show Descriptions (default)**
  - Each guide step has a description of what the step achieves.
  - Each parameter has a description of how it affects your project.
- **Hide Descriptions**
  - Step description and parameters description aren't displayed.

To switch between Wizard mode and Full view mode, click the Settings icon and switch the **Wizard mode** toggle.

To switch between Show Descriptions and Hide Descriptions mode, click the Settings icon and switch the **Show Descriptions** toggle.



You can combine the modes as you like, such as using Wizard mode with Hide descriptions mode. That way, you can see each step separately, but without any descriptions of steps or parameters.

Change the Code

You can make the code change by using Copy/Insert Snippet functionality. A code snippet provides a sample code that needs to be implemented in the respective file described in the guide step. In most cases, you need to provide input for the dynamic content that is substituted in the snippet. Once all the values are supplied, you can see the following options:

- **Copy**. Copy the code snippet to the clipboard. You can then insert it into any file and change it as you wish.
- **Insert Snippet**. Automatically inserts the code snippet into the relevant file. The applied change is highlighted, and the confirmation message “Code snippet has been successfully applied” appears.

ⓘ Note

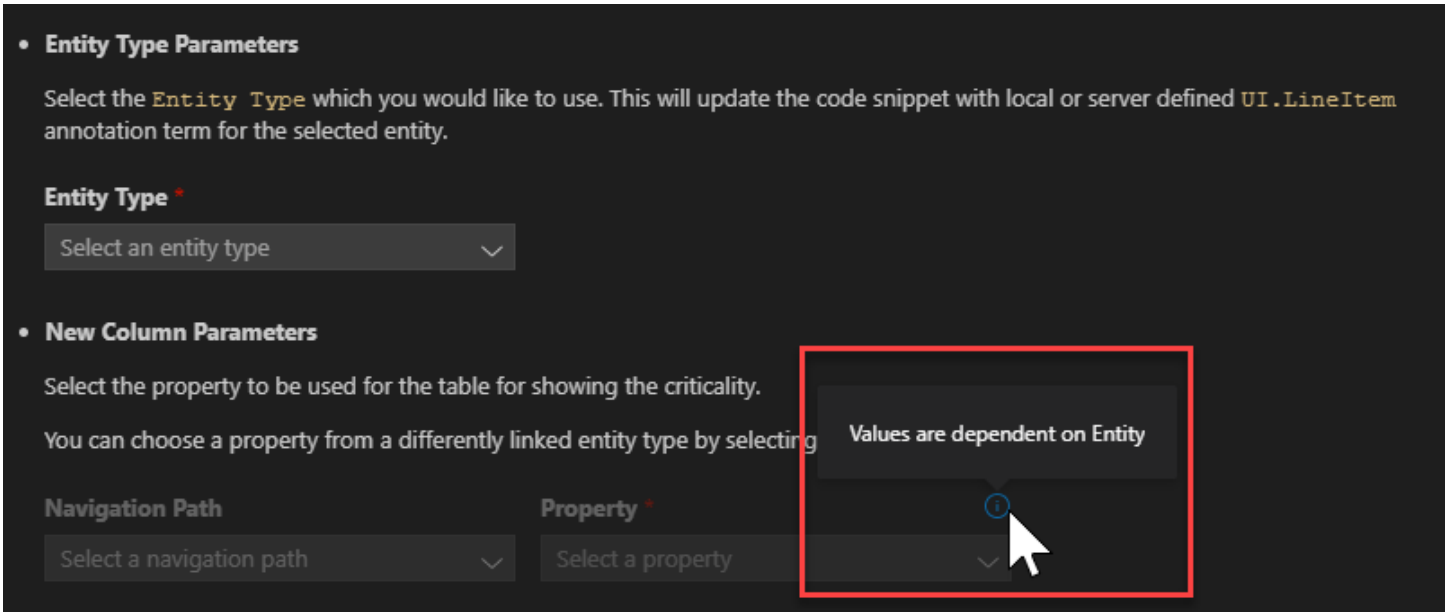
The **Insert Snippet** button is disabled if the guide can't be applied. It can happen if the guide refers to a different OData version or page type than the selected project, or if there's no project selected.

Parameters in Guides

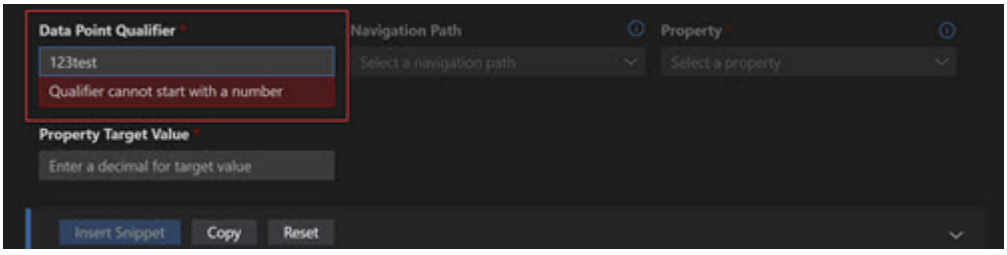
In some guides, code snippets don't have any parameters. Other guides contain drop-down lists and text boxes allowing you to specify the parameters to customize the code snippet.

Most drop-down lists parameters are context-dependent and only populated when Guided Development has context. Also, there are some static drop-down parameters, when the values in them remain the same regardless of the context.

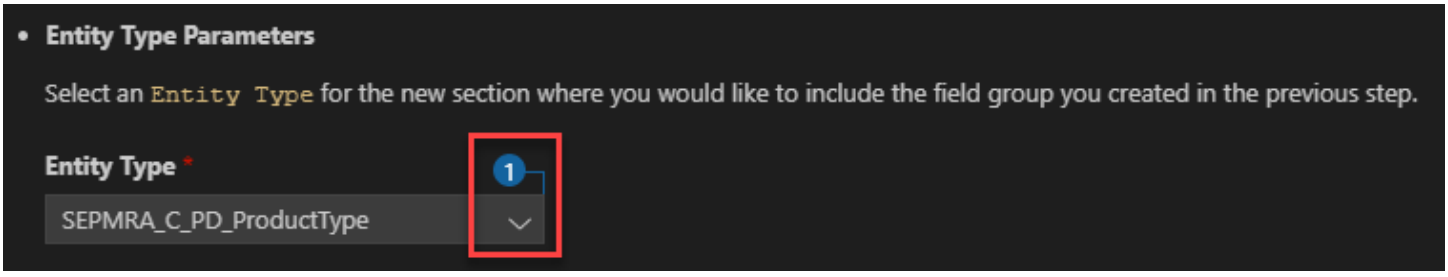
Some parameters depend on each other. For example, the Navigation Property parameter in the **Add semantic highlights to line items in tables based on their criticality** guide can only be selected after a value is selected for the Entity parameter. Parameters that are dependent on a previous parameter selection will be marked with an info icon tooltip. Hover over the icon to see where another parameter selection is required.



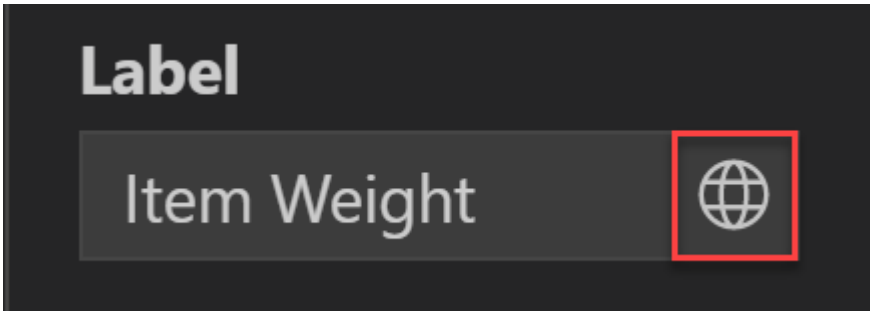
Parameters can be mandatory and optional. Mandatory parameters are marked with an asterisk symbol at the end of their name. If you click Insert Snippet without providing values for mandatory parameters, you'll see an error message for each mandatory field that isn't filled in. Some parameters also feature inline validation in the form of an error message to help you correct formatting issues. The error will be cleared and the Insert Snippet button will be reactivated once the issue is addressed.



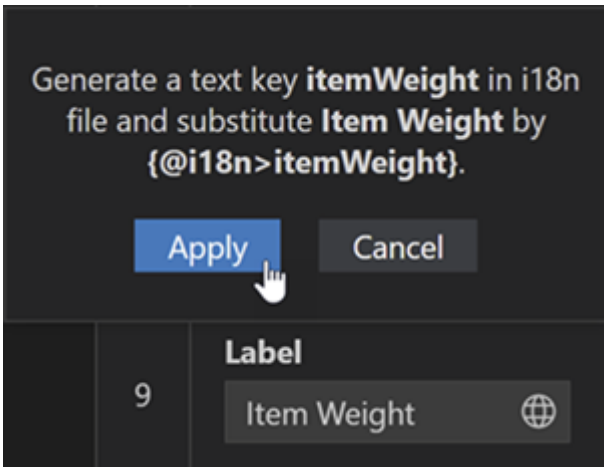
In addition, in some guides, such as **Add progress indicator column to a table** or **Add a field group as a section to a page**, parameters selected in the first step are prefilled in step 2 when appropriate. For example, the values selected for the Entity type parameter in step 1 will be filled as the default value for the Entity type in step 2 with the tooltip "1" displayed. If you want to change the prefilled values, you can do so manually.



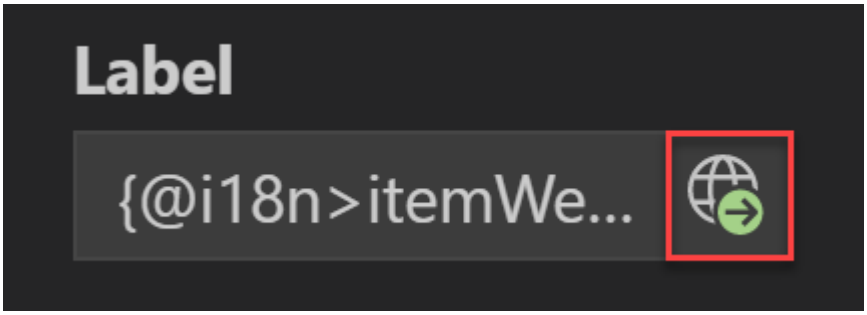
i18n keys for the globalization of your project can be automatically generated from the values you enter into input fields in guides. To create an i18n key in Guided Development, enter a value into the input field and press the Internationalization button.



A popup opens, asking if you would like to generate a text key for your value. Click **Apply**.



This will add the i18n key to your `i18n.properties` file. You can click the updated Internationalization button to jump to the file for further revisions if necessary.



Request a New Guide

The guides available in the Guided Development extension present the most commonly used features. Whereas, there are many more features available to the Fiori elements developer. For that reason, Guided Development also allows you to submit a request for a guide.

You can request a new guide by using one of the following options:

- Scroll to the bottom of the guides list and click **Request Guide** in the “Request new guide” section.
- Enter ***request guide*** to the “Search guides” field.
- Navigate your pointer to the Help icon and select **Request New Guide** from the drop-down list.

How to use the **Request Guide** form:

1. Fill in your name, email, title for a new guide, and description of what you need.
2. Click **Submit**.
3. When an email client opens, check your message and click **Send**.

For more information about features available in the SAP Fiori elements application, see Developing Apps with SAP Fiori elements.

Working with Annotations

Overview of Annotations

Local annotation files are created directly in the project. When you generate a project using SAP Fiori tools, one local annotation file gets generated in the `webapp/annotations` folder and is automatically registered in the `manifest.json` file. You can also create additional annotation files as needed using the annotation file manager or Maintaining Additional Elements.

The service metadata and backend annotation files (if any) are copied and stored in the `webapp/localService` folder during the project generation. The reference to these local copies are maintained in `manifest.json` using the '`localUri`', whereas the '`uri`' parameter contains the relative path to these sources in the back end. See Visualizing Annotations with Service Modeler about backend annotation.

It is important to define new annotation terms in the local annotation file of your project, as all the changes made in local copies of backend annotation file or metadata won't have an impact on the deployed application. If you need to change the annotation defined in the backend, you need to copy this annotation along with the target and qualifier to the local annotation file and adjust it there, see **Overriding backend Annotations** section. The annotation terms defined in the local annotation file will always win over the same annotation with the same qualifier and applied to the same target in the backend sources. The overriding sequence of multiple local annotation files is defined in the `manifest.json` file and can be viewed and changed in the annotation manager. For more information, see Visualizing Annotations with Service Modeler - **How to change the hierarchy of local annotation file** section.

You can either maintain your local annotations using the language server or you can use the support in Page Editor for a more schematic view. See Maintaining Additional Elements for more information.

ⓘ Note

Maintaining local annotation files with Service Modeler is only applicable to OData service, not CAP CDS.

Maintaining Annotations with Language Server

CAP CDS Files

Maintaining OData annotations in .cds files is accelerated by the SAP Fiori tools - CDS OData Language Server comprised in SAP CDS Language Support plugin. It assists you with adding and editing OData annotations in CDS syntax with:

- Code completion for annotations applied to entities and entity elements
- Validation against the OData vocabularies and project metadata
- Navigation to the referenced annotations

- Quick view of vocabulary information
- Internationalization (i18n) support for language dependent strings

See CAP CDS and Serving Fiori UIs: Adding Fiori Annotation for more information about CDS OData Language Server.

ℹ Note

SAP Business Application Studio - Fiori tools Dev space does not include CDS OData Language Server extension.

XML Annotation Files

Maintaining OData annotations in `annotation.xml` files is accelerated by the XML Annotation Language Server extension of SAP Fiori tools. It assists you with adding and editing OData annotations in XML syntax with the code completion, validation and other assisting features, same as CDS OData Language Server in CAP CDS files. To get this assistance, just open the local annotation file in the code editor. You can either open local annotation file from Service Modeler or single/double-click on the local annotation file of your project: `/webapp/annotations/<filename>.xml`.

Using XML Annotation Language Server

Prerequisites

To maintain the annotations using XML Annotation Language Server features, project needs to meet the following criteria:

- OData service

  Your project contains the local copy of service metadata. The path to this copy is provided in theSAP OData vocabularies `manifest.json` file as a local Uri.

  ### ℹ Note

  The local copies of the metadata and backend annotations are used for code completion and diagnostics in local annotation file, it is important that it stays in sync with the service metadata state in backend. It is not synced automatically with the metadata on the backend system. You can sync the local copy of the service metadata with the backend, see Syncing Annotations.

  The metadata of the OData serviceSAP OData must include one or multiple *< edm:Schema>* definitions within the *<edmx:DataServices>* element.

  According to the OData CSDL, your metadata file must contain a single EntityContainer.

  ### ℹ Note

  The namespace of the OData service should not contain / (slashes). The OData specification requires namespaces to consist of one or more SimpleIdentifiers separated by dots. Slashes are not supported. A SimpleIdentifier must start with a letter or underscore, followed by a maximum of 127 letters, underscores and digits.

- Local Annotation File

  Your project contains at least one valid annotation XML file that includes the *< /edmx:DataServices/Schema>* node and references to the metadata. The metadata namespace must match that of the metadata file.

  ### ℹ Note

  If your project does not contain an `annotation.xml` file or you need more than one, you can create a new annotation file automatically registered in `manifest.json`, see Visualizing Annotations with Service Modeler.

- `manifest.json` file

  Your project contains a `manifest.json` file with the following information:

  - Uri and localUri of the OData service
  - List of OData annotation sources
  - @i18n model with the uri to the `i18n.properties` file

  Note

  All paths used in annotation file are relative to the location of the `manifest.json` file

  Note

  For an overview of how to maintain the configuration in the `manifest.json` file, see Visualizing Annotations with Service Modeler.

Supported Vocabularies

The XML Annotation Language Server is based on the official OASIS vocabularies and https://github.com/SAP/odata-vocabularies (OData version 4.0) The following vocabularies are supported:

OData org

- Aggregation
- Authorization
- Capabilities
- Core
- Measures
- Repeatability
- Temporal
- Validation

SAP

- Analytics
- CodeList
- Common
- Communication
- Graph
- Hierarchy
- UI
- ODM
- Personal Data
- Session

OData

For more information about OData

- OData specifications

Limitations

- Annnotations directly embedded in the metadata are not supported
- Dynamic expressions are not supported

Visualizing Annotations with Service Modeler

You can view backend annotations in the SAP Fiori tools - Service Modeler, and maintain them with XML Code Editor.

Only OData service and CAP service annotations are supported. Annotations are associated to: projections, entities, and properties and are identified by the annotation icon in the entity or projection detail panel, or on the entity or projection node in the **Expanded View** and **List View**.

You can view the annotations associated to an entity, projection, or properties displayed in the Service Modeler **Annotation List View**.

For OData service, the Annotation **List View** displays the annotations associated to the target entity in both the backend and local annotations files. The annotation terms defined in the local annotation file win over the same annotation with the same qualifier and applied to the same target in the backend sources. Backend annotations can't be edited via the SAP Fiori tools - Service Modeler tool but backend annotations can be overridden in the local annotation file and edited manually via a **text editor** or using the XML Code Editor. The local annotations file is located: `/webapp/annotations/<filename>.xml`, see XML Code Editor for more information about how to work with local annotations.

Launching Service Modeler for Annotations

Service Modeler can be launched in several ways.

Using Command Palette

- Open **Command Palette**
- Start typing ***Service Modeler***
- Select **SAP Fiori tools: Service Modeler: Open Service Modeler**
- Select SAP Fiori elements project from your workspace.

Using folder context menu

If you already have a SAP Fiori elements project in your current workspace, you can right-click on any folder in your project and **Open Service Modeler**.

From the Text Editor

If your `metadata.xml` file is open in the text editor, click on the annotations icon .

Using Service Modeler for Annotations

To view annotations associated to a projection.

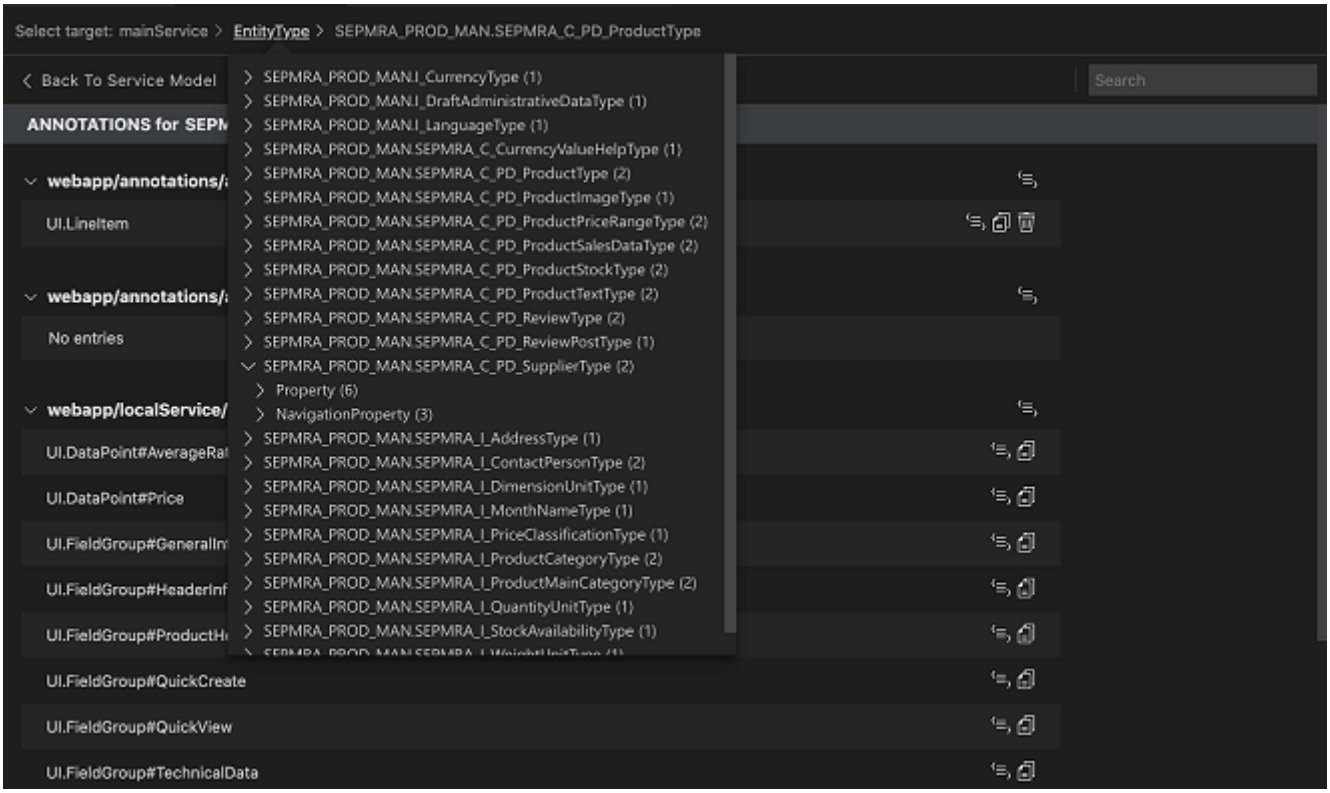
1. In any of the views, select the target projection.
2. Click on the **annotation** icon associated to the target projection or the property of the target projection in the project detail panel.
3. Click on **show source** icon beside the annotation. The source file is opened to the side with the annotation hightlighted.

Searching for Annotations

1. Select the target entity.
2. Click **annotation** icon.
3. Enter search criteria in the search input box in the upper-right corner.

Changing Target

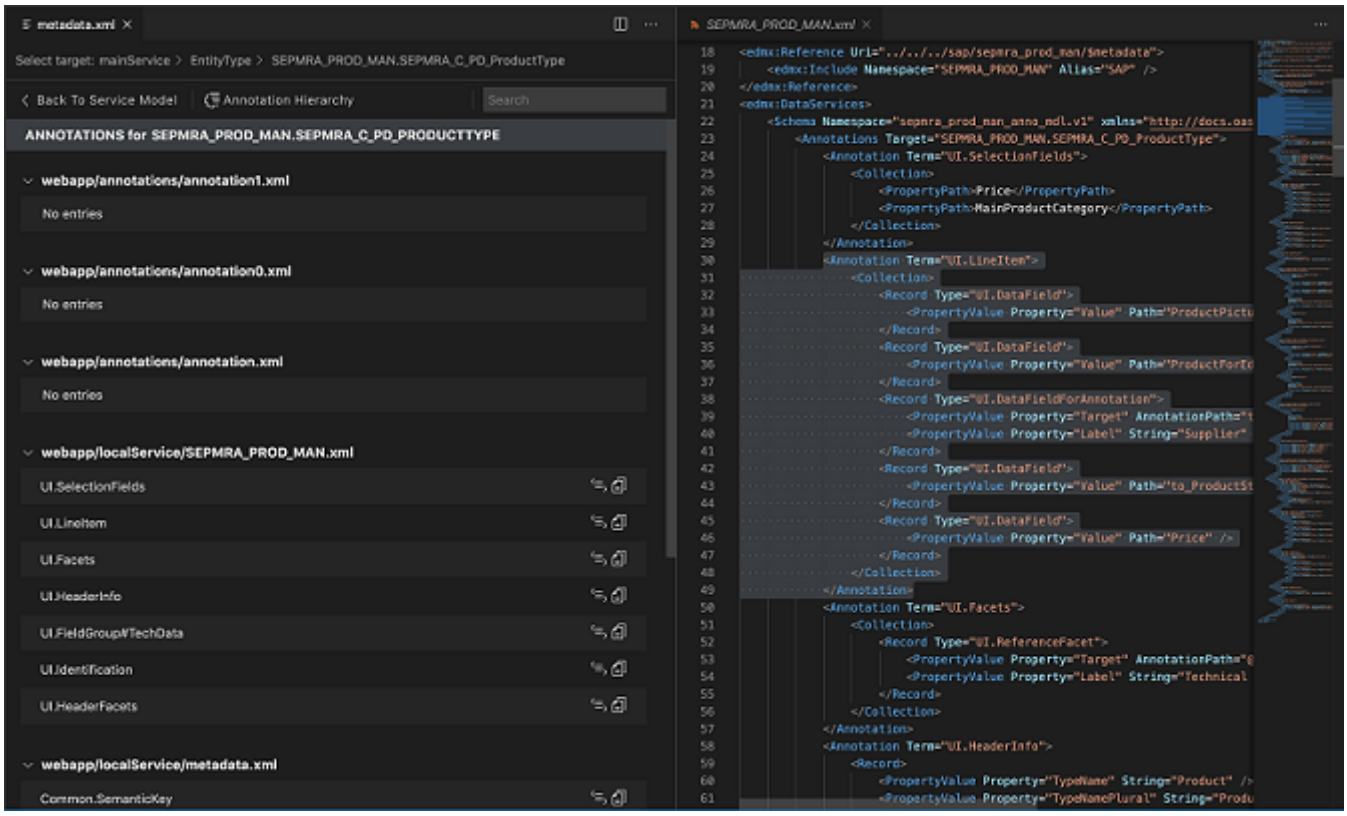
1. Ensure you are in **List View**.
2. Click the **select target** breadcrumb.
3. Select a service, entity, or property.



Editing Annotations

You cannot edit backend annotations with the SAP Fiori tools - Service Modeler. However, you can edit local annotations by navigating to XML Annotation Language Server with the Service Modeler.

1. Select the target entity.
2. Click the **annotation** icon.
3. Click the **go to editor** icon  opposite the selected local annotation in the service local annotation file.



4. Edit the local annotation manually in the XML Code Editor and save the local annotation file.

Deleting Annotations

1. Select the Target Entity.
2. Click the **annotations** icon .
3. Click the **delete** icon opposite to the selected local annotation in the service local annotation file.

Overriding Annotations

The following is a list of steps of how to override backend annotation to your local annotation file.

Overriding backend Annotations

1. Once you have found annotation of interest and are viewing backend annotation details.
2. Click **copy** icon opposite the **backend annotation**. This gives you a starting point for extending or customizing your application. Use Maintaining Annotations with Language Server to maintain your local annotation file.

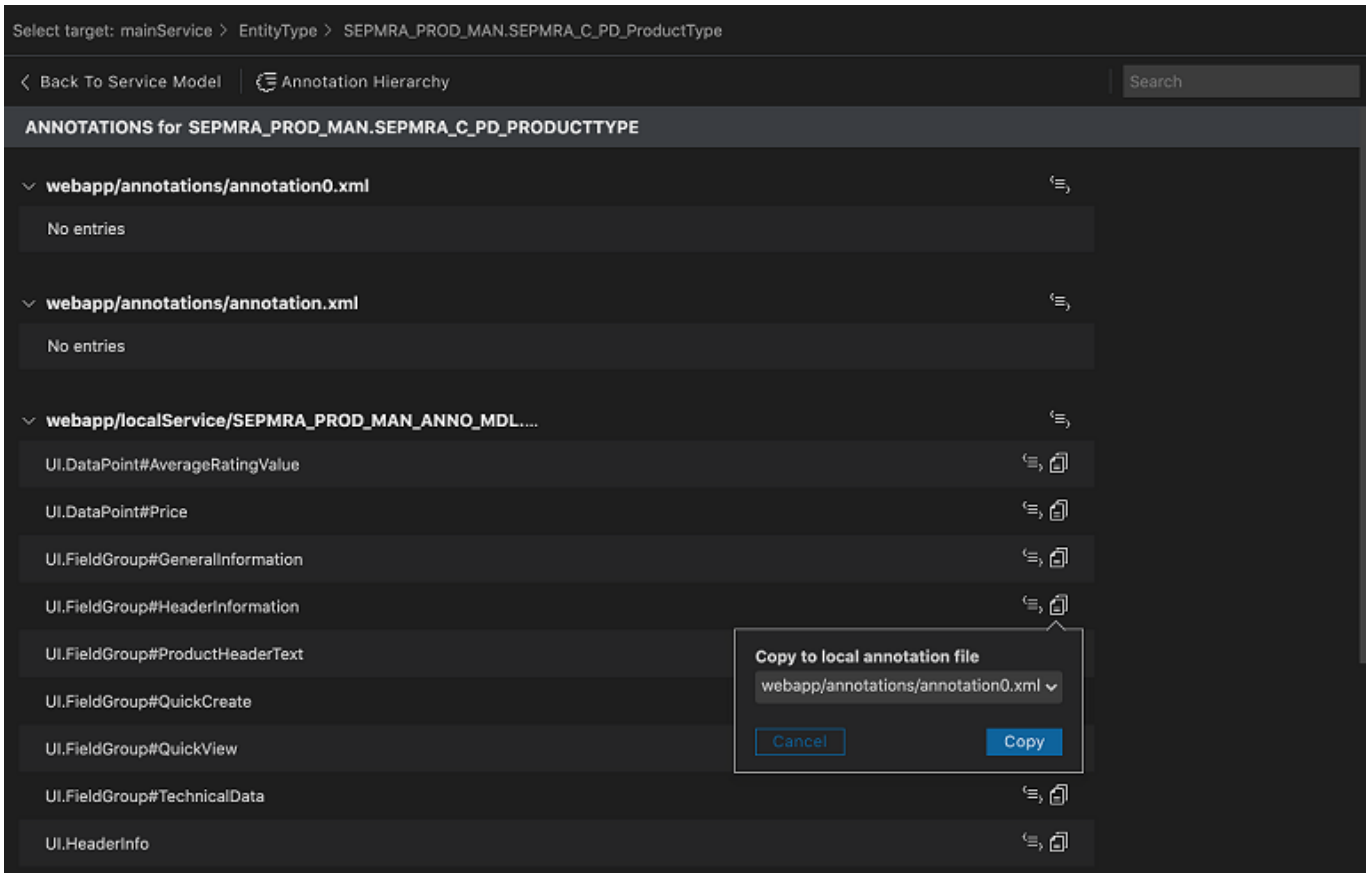
   If there's only one local annotation file associated to the service in the project, the annotation is copied automatically to that local annotation file. If there are multiple local annotation files associated to that service available in the project, you can choose which local annotation file you would like to copy the annotation to via a selection dropdown.

ℹ Note

You can add multiple local annotations files to your project.

ℹ Note

You can copy **local annotations** to and from **local annotation** files. You can't copy **local annotation** to **backend annotation** files.



Creating Additional Annotation Files

1. Right click on `manifest.json` file, select **Open Annotation File Manager**.
2. Click **Create Local Annotation File**.
3. Enter file name, namespace, and select the intended file folder location.
4. Click **Create**.

ⓘ Note

Additional annotation files can only be created for OData service, not CAP CDS.

Deploy an Application

Deployment Overview

SAP Fiori tools support deployment to ABAP as well as deployment to Cloud Foundry (CF) on SAP Business Technology Platform. In contrast to the SAP Web IDE approach, the deployment target isn't required during the application generation which allows a simplified user experience when the project is first created. This approach means that the decision could be deferred until when the developer is ready to deploy the application.

Deploying to ABAP

Prerequisite. Ensure all prerequisites are met when using an OData service to load data to the SAPUI5 ABAP repository.

For deployment to ABAP, the SAPUI5 Repository service exposed by the ABAP system is used to upload a deployment artifact. The ABAP backend provides all functionality to run the application, such as hosting, routing, and authentication. As a result, the deployment artifact is just an SAP Fiori application. In other words, it's a zipped dist folder of the SAP Fiori tools project.

For more information see, Deployment to ABAP

Deploying to SAP Business Technology Platform Cloud Foundry

Similar to the SAPUI5 Repository service in ABAP, SAP Business Technology Platform offers an HTML5 Repository to upload and host application. To access the HTML5 Repository, it is required to create an instance of an HTML Repository service. All other functionality required for running the application in an ABAP system comes out of the box and needs to be made accessible by creating instances of the corresponding services.

Deployment Process

The process to deploy the application involves the following steps:

- Generate deployment configurations
  - Generate Deployment Configuration ABAP
  - Generate Deployment Configuration Cloud Foundry
  - SAP Fiori Launchpad Configuration
- Deployment of Application

Troubleshooting Tips

The backend system contains the SAP_UI component version 7.53 or newer, but the SAPUI5 repository service can't be reached.

- Check if the service is activated. For more information, see Using an OData Service to Load Data to the SAPUI5 ABAP Repository.

The SAPUI5 repository service is active and reachable but whenever I deploy an application, I see the following error **`Request failed with status code 400`**.

- This could have multiple reasons, check the console for more information, or open transaction `/IWFND/ERROR_LOG` and check the server logs. A common issue is that during the setup, configuring a virus scan profile is forgotten. This can be corrected in the transaction `/IWFND/VIRUS_SCAN`.

ℹ Note

You can retrieve more detailed logging information when deploying to ABAP by executing the following commands during deployment.

- For detailed log messages from the backend services during deployment:

  MacOs/Linux: `DEBUG=ux-odata-client npm run deploy`

  Windows: `set DEBUG=ux-odata-client & npm run deploy`

- For detailed log messages from archiving and deploying the artefacts:

  MacOs/Linux: `DEBUG=ux-odata-client npm run deploy`

  Windows: `set DEBUG=ux-odata-client & npm run deploy`

ℹ Note

For any issues, create an incident in **SAP Support Portal** for the component **CA-UX-IDE**.

SAP Fiori tools CLI help

SAP Fiori tools CLI offers additional help options to find more information about the various commands and their options for deployment plus other commands.

- Execute the following command to list all the available commands: *npx fiori help*.
- Execute the following command to get the details about a specific command. For example, deploy: *npx fiori deploy help*.

Deployment Configuration

Configuration Options

In addition to defining parameters in the `ui5.yaml` file, every parameter can also be defined as environment variable that is referenced in `yaml`. Using the `dotenv` module, the task also supports project-specific environment variables defined in the `.env` file in the root of your project. Use the pattern `env:VAR_NAME` to reference an environment variable.

target

The target object contains properties identifying your target SAP system.

url

- `<string> pattern <protocol>://<hostname>[:<port>]` **(Required)**
- This parameter must contain a url pointing to your target SAP system.

client

- `<number> range [0..999]` **(Optional)**
- The client property is used to identify the SAP client that is to be used in the backend system. It translates to the url parameter `sap-client=<client>`. If the client parameter isn't provided, the default client is used.

params (Optional)

- `<string>` (optional)
- Specifiy addtional query paramaters to pass to the backend deployment `API`. It translates to the url parameter e.g. `sap-language=<2-digit sap language code>`. If the parameter isn't provided, the backend/user default is used.

scp

- `<boolean>` (default: `false`)
- By default the deployment task uses basic authentication when connecting to the backend. If the target system is ABAP Environment on SAP Business Technology Platform, this parameter needs to be set to `true`.

service

- `<string>` (default: `/sap/opu/odata/UI5/ABAP_REPOSITORY_SRV`)
- Path pointing to the SAPUI5 ABAP repository OData service in your target system. This parameter only needs to be used if the service is exposed at a different path in your backend system, for example, via alias.

credentials (optional)

The credentials object is required for `CI/CD` based deployments and it needs to contain the required parameters to authenticate at your target system. We strongly encourage to not add the credentials directly but use references to environment variables example `env:MY_VARIABLE` here.

For local usage (not in SAP Business Application Studio), we do not recommend to use the credentials object at all. As a result, the deployment task utilizes the operating systems secure storage maintain credentials.

username

- `<string>` **(Required)**
- SAP business user for the target system. The user requires authorizations to create/update the target ABAP development object.

password

- `<string>` **(Required)**
- Password required to authenticate the previously configured user. **IMPORTANT**: while technically possible to add the password to your configuration, we strongly **DISCOURAGE** that but recommend instead the use of environment variables.

app

The app object describes the backend object that is created/updated as result of the deployment.

name

- `<string>` **(Required)**
- Unique name of the application. The name is used as a part of the application url as well as the name of the ABAP development object used as container for the app.

package

- `<string>` **(Required for new apps)**
- Name of an existing ABAP package that is used as parent of the deployed application. The parameter is required for the creation of the application in the backend. Any following deployment updating the application does not require the package parameter, that is ignored .

transport

- `<string>` **(Optional)**
- The transport parameter refers to a transport request number that is used to record changes to the backend application object. The property is optional as it is only needed if the package for deployments requires transport requests.

description

- `<string>` **(Optional)**
- Optional description added to the created application object in the backend.

exclude

- `<string[] array of regex>` **(Optional)**
- By default, the deployment task creates an archive (zip file) of all build files and sends it to the backend. By using exclude, you can define expressions to match files that shall not be included into the deployment.

> **Note**
>
> `string.match()` is used to evaluate the expressions.

index

- `true|false` **(Default: `false`)**
- If set to `true`, then an additional `index.html` is generated and deployed to run the application standalone.

Location of MTA Directory

The tool finds the nearest parent directory that contains `mta.yaml` and offers that as the MTA directory. Failing that, it defaults to the parent directory of the application.

Destination

Destination configured to connect to the backend on Cloud Foundry. If there's a setting in `ui5.yaml`, that value is offered as the default.

Prefix

Prefix is used for the ID of the MTA and the service names. It defaults to the namespace of the app. If a namespace is not found, it defaults to `test`. Select a prefix so that the service names are unique to your MTA. Otherwise, deployment by multiple people will overwrite the same service. At the end of the generation, it is possible to optionally generate SAP Fiori launchpad configuration (default: no).

(Optional): Setting environment variables in an .env file

If you prefer to keep the environment variables in a file, you can create the `.env` file at the root of your project that contains the environment variables to be referenced in the `ui5-deploy.yaml` file.

We recommend that you do not have your actual username and password in the `ui5-deploy.yaml`. In this case, you will be asked to provide these credentials during deployment if needed.

> **Sample Code**

```
XYZ_USER=[MY_USER_NAME]
XYZ_PASSWORD=[MY_PASSWORD]
```

Generate Deployment Configuration ABAP

ABAP Pre-Requisites

The deployment to ABAP task allows deploying SAP Fiori applications to SAP systems using the SAPUI5 Repository OData service.

Prerequisites:

- SAP component SAP_UI 7.53 or higher is installed in your SAP system

  > **Note**
  >
  > For systems below 7.53, the alternative is to upload the application.

- Service needs to be enabled and accessible from your development environment (activate and maintain services)
- For operations on a SAPUI5 ABAP repository, you need the S_DEVELOP authorization

Limitations

- The task doesn't create ABAP transports, therefore, it requires an existing transport if the target ABAP package requires a transport
- Basic Authentication (user/password based authentication) is supported for all backend systems. Additional support for OAuth2 authentication is provided for ABAP systems on SAP Business Technology Platform.

Generation of Deployment Configurations

In order to create deployment configuration, launch the deployment configuration wizard from the command palette entry `Fiori: Add Deployment Configuration` and chose the **Fiori** project you would like to configure, or you can launch from the command line using the command `npx fiori add deploy-config` whilst in the required **Fiori** project folder.

You're prompted for required information and then the `ui5-deploy.yaml` file is created based on your input and the content of the existing `ui5.yaml` file used for preview. In addition to creating the configuration, the create deployment command will also update your `package.json` so that you can execute `npm run deploy` - afterwards to deploy your application. See Deployment of Application.

When prompted, add or choose:

Please choose the target	ABAP
Select Target System	Choose a system from your SAP saved systems or provide a Target system URL(VS Code only).
Destination	Choose the deployment destination from list provided (SAP Business Application Studio only).
Enter client	Add a new client or leave **default**.
SAPUI5 ABAP Repository	Add a name for the deployed application.
Deployment Description	Add the optional description for the deployed application.
Package	Add a valid package name.
How do you want to enter Transport Request	• **Enter manually:** Manually provide the transport request. • **Choose from existing:** The applicable list of transport requests will be retrieved from the target system and displayed in a list for you to choose. If the list of transport requests is unable to be retrieved from the target system, user must provide the entry manually. • **Create new:** A new transport request is automatically created for use. If the transport request is unable to be created from the target system, user must provide the entry manually.
Transport Request	When, prompted either choose a transport request from the list or add a valid transport request.

Example Configuration - ABAP

Executing `ui5 build --config ui5-deploy.yaml` in your project with the configuration below in a `ui5-deploy.yaml` manually added to the project, would deploy all files of your `dist` folder except files ending with `.test.js` and the `internal.md` file. The target system is XYZ with client 200. Username and password for authentication is read from the environment variables XYZ_USER and XYZ_PASSWORD.

Based on this example, the application is created/updated as `/TEST/SAMPLE_APP` in package `/TEST/UPLOAD` and all changes is recorded in transport request `XYZQ300582`.

Sample **content of `ui5-deploy.yaml`**

Sample Code

```
builder:
  customTasks:
  - name: deploy-to-abap
    afterTask: replaceVersion
    configuration:
      target:
        url: https://XYZ.sap-system.corp:44311
        client: 200
        auth: basic
      credentials:
        username: env:XYZ_USER
        password: env:XYZ_PASSWORD
      app:
        name: /TEST/SAMPLE_APP
        package: /TEST/UPLOAD
        transport: XYZQ300582
      exclude:
      - .*\.test.js
      - internal.md
```

Example Configuration - Additional params

For example, you can add additional settings params such as `sap-language` to your yaml file.

Sample **content of `ui5-deploy.yaml`**

Sample Code

```
builder:
  customTasks:
  - name: deploy-to-abap
    afterTask: replaceVersion
    configuration:
      target:
        url: https://XYZ.sap-system.corp:44311
        client: 200
        auth: basic
        params:
          sap-language: en
      credentials:
        username: env:XYZ_USER
```

```
  password: env:XYZ_PASSWORD
app:
  name: /TEST/SAMPLE_APP
  package: /TEST/UPLOAD
  transport: XYZQ300582
exclude:
- .*\.test.js
- internal.md
```

Generate Deployment Configuration Cloud Foundry

Cloud Foundry Prerequisites

- MTA **executable in the path**.

  For SAP Fiori tools application deployment to Cloud Foundry, you must install MTA. It's a tool for exploring and validating the multitarget application descriptor (`mta.yaml`). The tool can be used as a Go library or as a command-line tool, also available as an npm package.

  You can install it globally by running the following command:

  ```
  npm i -g mta
  ```

  - **Mac users**. When installing MTA, the following error may occur:

    ```
    Error: EACCES: permission denied, mkdir bin.
    ```

    .In this case, permission needs to be altered by using the following command:

    ```
    sudo chown -R $(whoami) FOLDER-NAME
    ```

  - **Windows users**. To build an MTA archive from a project source code in a Windows environment, install `GNU Make 4.2.1` on your machine. If required, you can install `Make` by following the instruction from Cloud MTA Build Tool .

  For information on adding a project to MTA config, see SAP Fiori Elements.

- **Cloud Foundry CLI tools**.

  To access SAP Business Technology Platform, download and install `CF CLI`, which is the official command line client for Cloud Foundry, from the Cloud Foundry CLI page . We recommend that you follow the installation instructions when installing `CF CLI` tools.

  - **Windows users**. You also need to create a `CF_HOME` variable. To do so, perform the following steps:

    1. Navigate to **Environment Variables**.
    2. Click "New" under the user variables.
    3. Enter `CF_HOME` as a variable name.
    4. Set write permission for a directory.
    5. Click **OK**.
    6. Close all command line windows.

- **MultiApps Cloud Foundry plugin**.

  MultiApps Cloud Foundry plugin performs multitarget application (MTA) operations in Cloud Foundry, such as deploying, removing, viewing, and more. To install the Multi-AppsCloud Foundry CLI plugin, execute the following command:

  ```
  cf install-plugin -r CF-Community "multiapps"
  ```

We recommend that you follow the installation instructions when installing MTA plugin.

For more information on Cloud Foundry plugins, see https://plugins.cloudfoundry.org .

- **A correctly configured destination to the back-end system**.
- **User authorization on Cloud Foundry to deploy**.

  To connect to Cloud Foundry, execute

  Code Syntax

  ```
  cf login -a https://api.cf.sap.hana.ondemand.com
  ```

Generate Deployment Configuration Cloud Foundry

For the deployment to Cloud Foundry, an MTA configuration will be created. The command allows to create a new configuration i.e. a new `mta.yaml` file or updates an existing `mta.yaml` with the information required for deployment. After successfully creating the configuration, running `npm run build` in the MTA directory that contains the application will try to build a deployable mtar file that can then be deployed to CF with `npm run deploy`.

To generate MTA project, perform the following steps:

1. Launch the deployment configuration wizard from the command palette entry `Fiori: Add Deployment Configuration` and chose the **Fiori** project you would like to configure, or you can launch from the command line using the command `npx fiori add deploy-config` whilst in the required **Fiori** project folder.
2. When prompted, enter or select:

Prompt	Entry
Choose Target	Cloud Foundry
Destination Name	If the destination name is empty or not the correct target, enter the destination name.

   Note

   Any instance-based destinations defined in the project `mta.yaml` file will be available as a destination option in SAP Business Application Studio. These destinations will be displayed with the label **Instance Based Destination** after the destination name.

3. Next, the installer runs and you can see updates in the project tree.

   Note

   If no `mta.yaml` file is found in the application folder, you will be given the option to create one during deployment configuration.

Artifacts & configuration created

Running the command/task results in a directory structure that looks similar to the following:

```
mta_directory
|_ application_directory
   |_ ...
```

```
    |_ webapp
       |_ ...
       |_ manifest.json
    |_ ui5-deploy.yaml
    |_ ui5.yaml
    |_ xs-app.json
...
|_ package.json
|_ mta.yaml
|_ xs-security.json
```

SAP Business Application Studio Cloud Foundry Support

If you don't have an MTA in your dev space in SAP Business Application Studio, you can use the provided generators to create the required configuration as follows:

1. Open File + New Project from Template .
2. Select **Basic Multitarget Application**.
3. Enter a project name, and then click Finish.

The IDE opens again with MTA as a workspace. You can then add app router configuration:

1. Right-click on the newly generated MTA file, and select **Create MTA Module from Template**.
2. Select **Approuter Configuration** and provide the relevant details and click Next.
3. As a result, the `mta.yaml` file is updated with the destination-content module. For full details on mta.yaml updates, please see Managed Approuter Project Result.

After the app router has been added, you can proceed to add your SAP Fiori project:

1. Right-click on the newly generated MTA file, and select **Create MTA Module from Template**.
2. Choose the SAP Fiori application generator.
3. Provide your application details.
4. By default, deployment configuration will be enabled and added to the existing MTA file after generation.

SAP Fiori Launchpad Configuration

Depending on deployment to Cloud Foundry or ABAP, you can add configuration to deploy the application to SAP Fiori launchpad.

Command: npx fiori add flp-config

It is possible to create configuration required to run the application in an SAP Fiori launchpad. This command updates the application `manifest.json` with the required inbound navigation property, required for integrating with the SAP Fiori launchpad.

Semantic Object	name<unique>
Action	display

Title	Title of an application
Subtitle (Optional)	Subtitle to be used by the tile

Deployment of Application

Deployment to ABAP

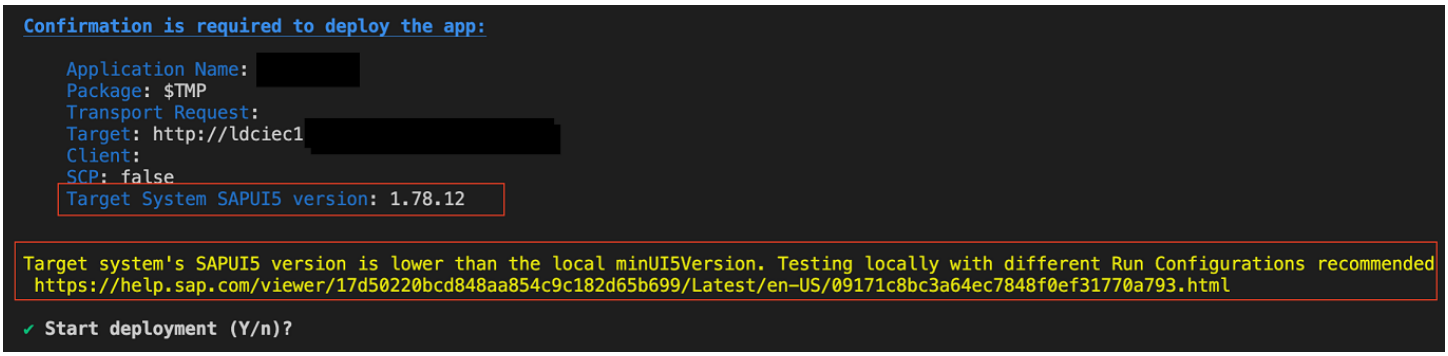
Deploy using the following command:

```
npm run deploy
```

When prompted, check deployment configuration and press `Yes` to proceed.

If authentication to the back end is required, a prompt for username and password appears.

To avoid any issues post deployment due to SAPUI5 version being lower in the target system, if applicable, you will see additional information in the deployment confirmation message about the target system's SAPUI5 version and recommendation to test your application more thoroughly using the Use Run Control.



Based on the sample configurations above, after a successful deployment, you should be presented with the URL for the deployed application in the generated logs. An example of the generated logs is as follows. You should be able to copy, or click on the URL in the logs to launch your application.

Sample Code

```
...info builder:custom deploy-to-abap * Done *
...info builder:custom deploy-to-abap App available at https://host:port/sap/bc/ui5_ui5/sap/app/
...
...info builder:custom deploy-to-abap Deployment Successful.
...info builder:builder Build succeeded in 18 s
```

Users can also choose to deploy an archive file to ABAP by using either of the following commands:

- ### Sample Code

  ```
  npx fiori deploy --archive-path 'somefile.zip'
  ```

  Where **somefile.zip** is valid ABAP archive.

- **Sample Code**

```
npx fiori deploy --archive-url 'https://someurl.com/archive.zip'
```

Where **https://someurl.com/archive.zip** points to a valid ABAP archive file that is accessible without authentication.

You can also append the following parameters to apply the required deployment configuration needed for the archive file:

Sample Code

```
'-d': 'destination',
'-u': 'url',
'-l': 'client',
'-t': 'transport',
'-n': 'name',
'-p': 'package',
'-e': 'description',
'-c' : '/path/to/ui5-deploy.yaml'
```

Deployment to Cloud Foundry

1. Connect to Cloud Foundry.
2. Deploy to Cloud Foundry:
    a. Navigate to a new root folder of the **mta** project. The folder containing the `mta.yaml` with

    ```
    cd ..
    ```

    b. Build the multitarget archive with

    ```
    npm run build
    ```

    c. Deploy to SCP with

    ```
    npm run deploy
    ```

    d. This process takes a few minutes.
3. Preview the deployed application - **Standalone Approuter**.
    a. In your SAP BTP Cockpit, select your target space.

    By default, the **Applications** tab is selected in the left pane and displays all the deployed HTML5 applications within this space.

    b. Use the **Search** box to search for a specific router name.
    c. Click the router name to open the **Overview** page.

    The deployed URL is displayed under **Application Routes**.

    d. Click the link to open the HTML5 application.
4. Preview the deployed application - **Managed Approuter**.
    a. In your SAP BTP Cockpit, select your target space.
    b. Select the **HTML5 Applications** tab in the left pane.
    c. Use the **Search** box to filter the list of HTML5 applications by the name of your application.

d. Click the application name to open the HTML5 application.

To retrieve the deployed URL using the Cloud Foundry CLI, open a new terminal and run the following command:

```
cf html5-list -u -di <mta-id>-destination-service -u --runtime launchpad
```

where `<mta-id>` is the `ID` field specified in your `mta.yaml`.

ⓘ Note

`<mta-id>-destination-service` is the name of the resource defined with a type `destination`.

ⓘ Note

To use the above command, the latest version of html5-list v1.4.6 is installed as follows:

```
cf install-plugin -r CF-Community "html5-plugin"
```

If you do not have access to the `mta.yaml` file or the project source code, you can list all HTML5 deployed applications for your Cloud Foundry Space, using the command `cf html5-list`.

Find the required project and replace `<mta-id>` with the name column field.

ⓘ Note

When using SAP Business Application Studio, you can right-click on the **mta** file and select Build MTA Project to create the deployment artefact. However, this creates a deployment artefact that cannot be deployed using the target

```
npm run deploy
```

Please ensure you build the multitarget archive with the following command.

```
npm run build
```

Deployment to ABAP system in test mode

If deployment target is an **ABAP system**, you can choose to deploy your application in **Test Mode**. **Test Mode** doesn't deploy your application, but will show the results of operations (create, read, update, delete) that would be done in a real run for each file to help you make an informed decision. A successful **Test Mode** execution does not necessarily mean that your upload will be successful. As a developer you can use this test mode to make sure your configurations are correct (including backend system) at any point during your app development.

1. Verify that your project's **`package.json`** contains **deploy-test** script:

   a. `deploy-test:fiori deploy --config ui5-deploy.yaml --testMode true`

ⓘ Note

Make sure in your `package.json`, `@sap/ux-ui5-tooling` version is at least **1.3.5** or higher.

2. To update the `@sap/ux-ui5-tooling`, use command `npm i @sap/ux-ui5-tooling@latest --save-dev`.
3. In the Terminal, navigate to the project folder. To deploy in test-mode,from within the project folder run `npm run deploy-test` using the script in the package.json

Undeploy an Application

SAP Fiori tools support undeployment from ABAP systems as well as undeployment from Cloud Foundry (also known as CF) on the SAP Business Technology Platform.

Undeployment from ABAP

To undeploy an application that is deployed to an ABAP system, perform the following steps:

1. In the terminal, within your project, execute the following command:

   ```
   npm run undeploy
   ```

2. When prompted, check undeployment configuration and press Y (Yes) for confirmation.

   → Tip

   If a `package.json` file does not contain an undeployment script, update `@sap/ux-ui5-tooling` and add the following script to `package.json`:

   ```
   "undeploy": "fiori undeploy --config ui5-deploy.yaml"
   ```

Undeploy an application from outside a specific project

To undeploy an application from outside a specific project, replace a placeholder and enter the command depending on the environment you work in.

- In VS Code, execute the following command:

  ```
  npx @sap/ux-ui5-tooling fiori undeploy --url <Target_ABAP_system_url> --name <Application_name
  ```

- In SAP Business Application Studio, execute the following command:

  ```
  npx @sap/ux-ui5-tooling fiori undeploy --destination <Destination_name> --name <Application_na
  ```

To find the correct values for each command, see the `ui5-deploy.yaml` file in your application.

Undeployment from Cloud Foundry

To undeploy an application from Cloud Foundry, perform the following steps:

1. Connect to Cloud Foundry:

   ```
   cf login -a
   ```

   For more information, see https://api.cf.sap.hana.ondemand.com/.

2. In the terminal, execute the following command:

   ```
   cf undeploy <mta-id> --delete-services --delete-service-keys
   ```

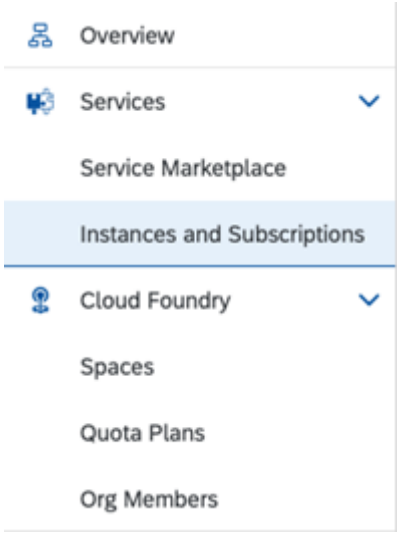
Expose Application to Central Application Router

Exposing an application to the central application router supposes the creation of subaccount level destinations requiring the organization manager authorizations.

Open the SAP Business Technology Platform Cockpit

Prepare Authentication Service

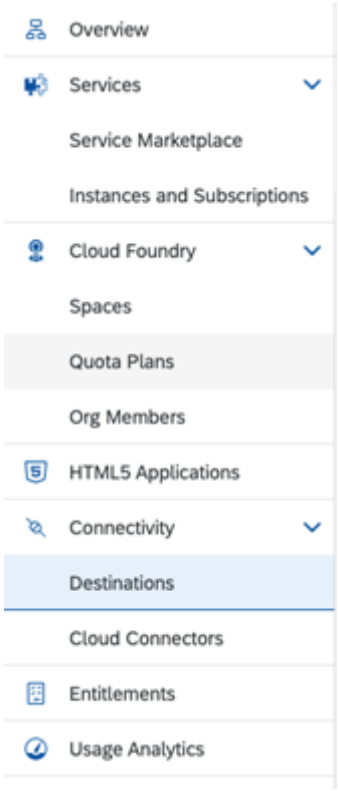
1. Open **Service Instances**.



2. Select **Authorization & Trust Management**.
3. Search for your XSUAA service e.g. type *test-<something_unique>*.
4. Click > on the right side of the row containing your service.
5. If there is no service key, create a new one (click three dots in the upper right corner).
6. Copy the name of a service key.

Expose Authentication Service

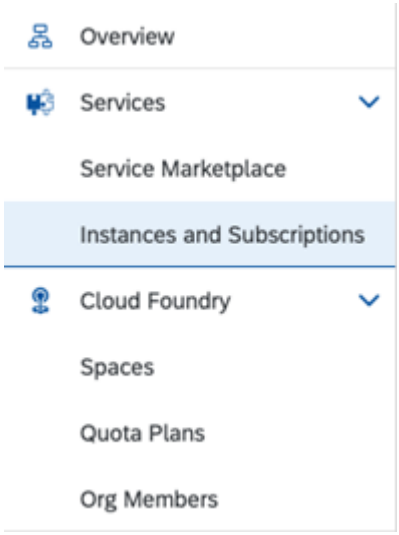
1. Open **Connectivity** > **Destinations**.



2. Click *New Destination*
3. Select *Service Instance*
4. *Service Instance:* Select the XSUAA service created with your **mta** e.g. *test-<something_unique>-uaa*
5. *Name:* Give it a meaningful name e.g. *test-<something_unique>-uaa*
6. Click on *Next*
7. Click on *New Property* and add *ServiceKeyName* and paste the name copied before e.g. *test-<something_unique>-uaa-service-key*
8. Click on *New Property* and add sap.cloud.service: *test-<something_unique>*
9. Click on *Save*

Prepare HTML5 Repository Service

1. Open Instances and Subscriptions.

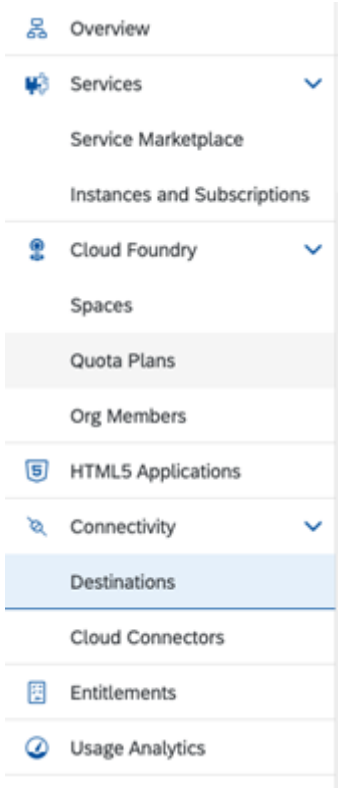


2. Select *HTML5 Application Repository*
3. Search for your service e.g. *type test-<something_unique>*
4. Click on the *>* on the right side in the row with the service with *Plan app-host*

5. If there is no service key, create one (click on three dots in the upper right corner)
6. Copy the name of a service keyCopy

Expose HTML5 Repository Service

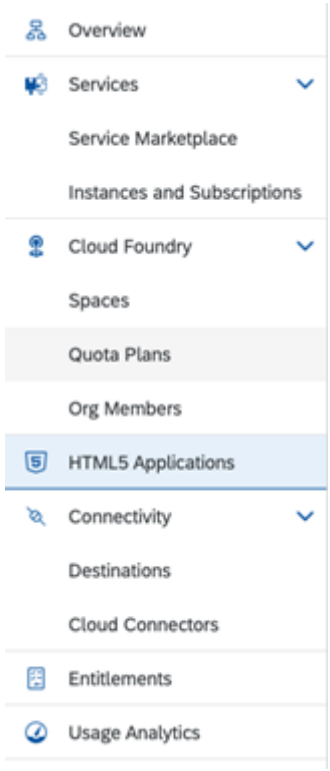
1. Open **Connectivity** > **Destinations**.



2. Click on *New Destination*
3. Select *Service Instance*
4. *Service Instance*: Select the HTML5 Repository service created with your mta e.g. *test-<something_unique>-html5-repo-host*
5. *Name*: Give it a meaningful name e.g. *test-<something_unique>-html5-repo-host*
6. Click on *Next*
7. Click on *New Property* and add *ServiceKeyName* and paste the name copied before e.g. *test-<something_unique>-deployer-<something_unique>-html5-repo-host-credentials*
8. Change the suggested property *sap.cloud.service* to *test-<something_unique>* (same as for xsuaa service destination)
9. Click on *Save*

Test Application

1. Open **HTML5 Application**.



2. Search for your app(s) *test-<something_unique>*
3. Click on your application *test<something_unique>ztravel*

Enable integration into cFLP

Open the `manifest.json` of your application and add the parameter below to the root node:

```
"sap.cloud": {
    "public": true,
    "service": "test-<something_unique>"
  }
```

Rebuild and deploy your mta project.

ℹ Note

If you do not execute the steps above then you will be able to configure the application to work in cFLP but it will fail with an error when being loaded.

Security

When deploying to a system please see Securing Apps section for additional information about security scans of SAPUI5 applications.