PUBLIC

Document Version: 2210 – 2022-10-13

Extensibility Guide

© 2022 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

Content

1 Extensibility Overview

This topic provides you with an overview about extensibility and the available extension.

Introduction

With extensibility, you can customize and enhance existing products and functionalities based on industry-specific or customer-specific requirements. This guide provides you with an overview of the available extension for the products within SAP Intelligent Asset Management and how to make use of this extension.

Extension for Asset Central UI Applications

With the extension for the asset central UI applications, you can create extensions for the following applications:

- Equipment
- Models
- Locations
- Spare Parts
- Systems
- Failure Modes
- Functional Locations

For more information, see Extension for Asset Central UI Applications [page 4].

2 Extension for Asset Central UI Applications

This section contain steps to create a new extension application in the SAP BTP, Cloud Foundry (CF) environment. For extension development, customers can use the SAP Web IDE service from the SAP BTP, Neo (Neo) environment until the service is available in the CF environment. Create a new Neo subaccount (Frankfurt region) if it has not already been created and enable the SAP Web IDE in that subaccount. This document covers two types of extensions:

- Asset Central Foundation (AC) Extensions
- Standalone Extensions

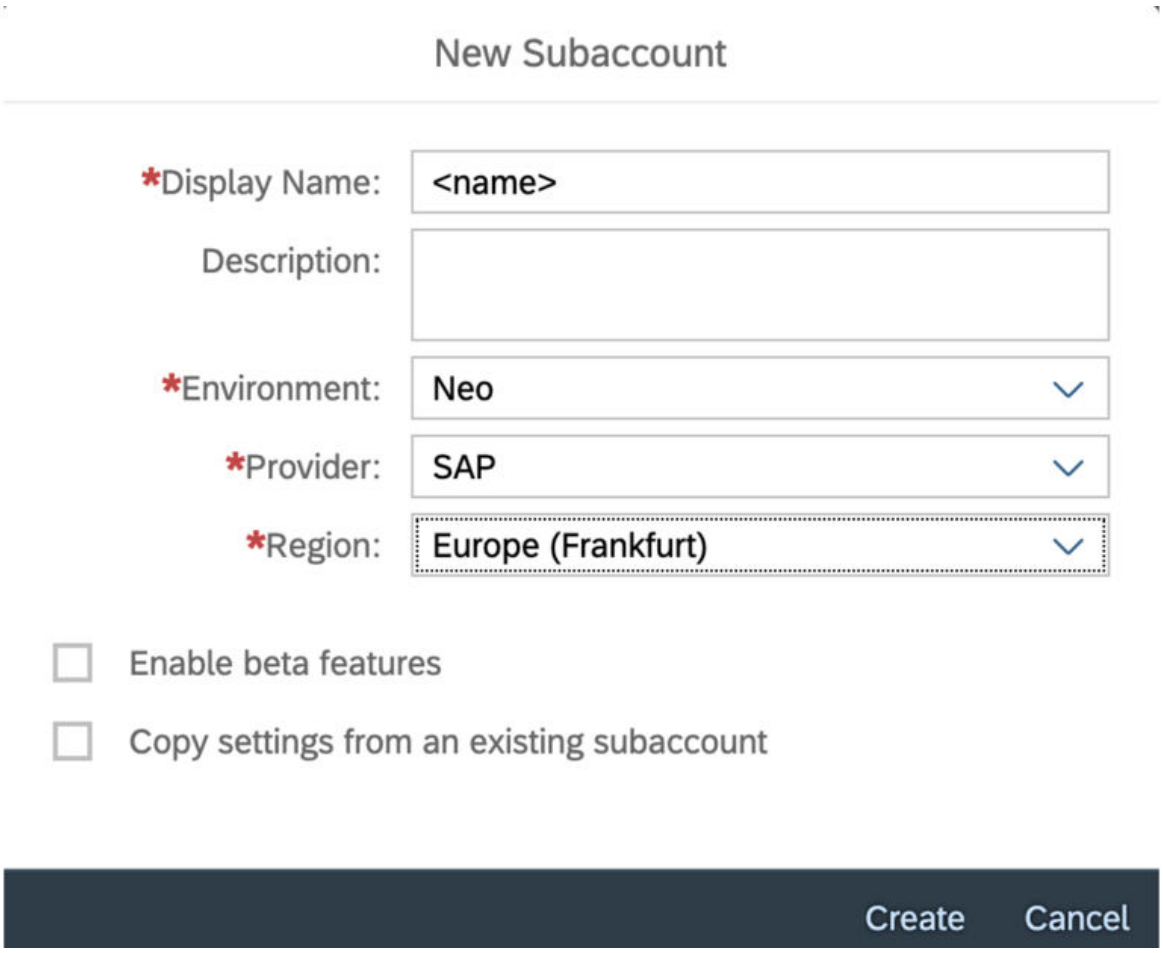
Before we proceed with extension development, it is important that we also configure the Cloud Foundry space we want to use for local testing. To configure the Cloud Foundry space, navigate to the preferences tab in Web IDE and select the Cloud Foundry section. In the Cloud Foundry section, configure the relevant space.

2.1 Creating Extensions for Asset Central Foundation Applications

The asset central foundation extensions plugin enables you to extend asset central foundation applications. To be able to access the asset central extension plugin, you need to create a plugin destination to your Neo account where you have the SAP Web IDE service enabled.

Prerequisites

1. Create a new Neo sub account (Frankfurt region).



2. Enable the SAP Web IDE service.

Related Information

https://help.sap.com/viewer/825270ffffe74d9f988a0f0066ad59f0/CF/en-US/995af3aa09074ceca20e59011ef78529.html

2.1.1 Creating a Destination for the SDK

This topic describes how to create a destination for the SDK.

Context

To activate the SDK in SAP Web IDE, you need to create a destination for the SDK so that it is visible in SAP Web IDE.

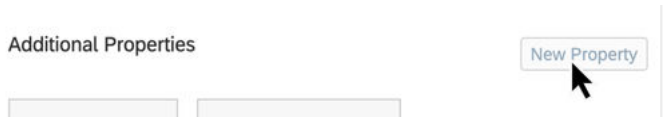
> **i Note**
>
> Web IDE is only available in the Neo environment.

Procedure

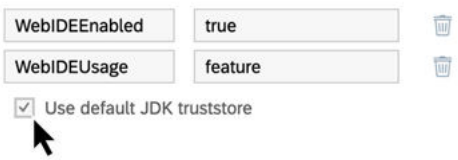
1. Go to the SAP BTP Cockpit.
2. Go to your subaccount as follows:
   a. In the left pane, choose *Regions* and choose your Neo environment.
   b. On the global accounts screen, choose your global account.
   c. On the subaccounts screen of the selected global account, choose your subaccount.
3. To create a new destination, choose *Connectivity* > *Destinations* > *New Destination*:
4. In the *Destination Configuration* window, fill in the following fields:

Name	Enter a name for the destination.
Type	Select *HTTP*.
Description	Enter a description to provide additional information.
URL	Enter the following URL: `https://iot-ain-live-iam-webide-templates.cfapps.eu10.hana.ondemand.com`
Proxy Type	Select *Internet*.
Authenticiation	Select *NoAuthentication*.

5. Add two properties as follows:
   a. Under *Additional Properties*, choose *New Property*:



   A new property with two fields is displayed.
   b. In the left field, select *WebIDEEnabled*.
   c. In the right field, enter **`true`**.
   d. To add the second property, choose again *New Property*.
   e. In the left field, select *WebIDEUsage*.
   f. In the right field, enter **`feature`**.
6. Activate the *Use default JDK truststore* checkbox if it is not activated:



7. Choose *Save*.

> **i Note**
>
> It takes up to 5 minutes until the destination is created.

Results

The destination is created and displayed in the list.

Related Information

Activating the SDK [page 7]

2.1.2 Activating the SDK

This topic describes how to activate the SDK.

Prerequisites

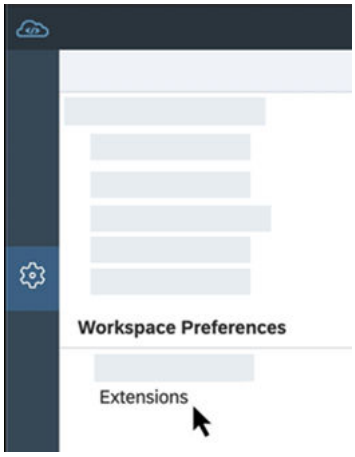
You have created a destination for the SDK. For more information, see Creating a Destination for the SDK [page 5].

Context

To generate an asset central foundation application based on the SDK templates, you need to activate the SDK in SAP Web IDE.

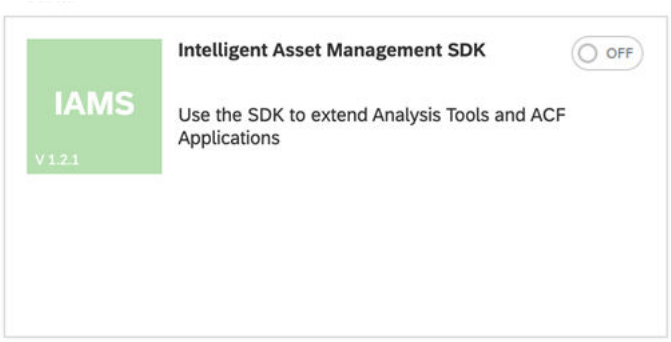
Procedure

1. Open SAP Web IDE. For more information, see Open SAP Web IDE.
2. Choose *Workspace Preferences* > *Extensions*:



3. In the search field, enter *Intelligent Asset Management SDK*.

   The Intelligent Asset Management SDK is displayed:



4. Turn on the SDK.

5. Choose *Save*.
6. Confirm the displayed query to refresh SAP Web IDE.

Results

The SDK is activated.

2.1.3 Generating an Asset Central Foundation Extension Application

This topic describes how to generate an asset central foundation application.

Prerequisites

You have activated the SDK. For more information, see Activating the SDK [page 7].

Context

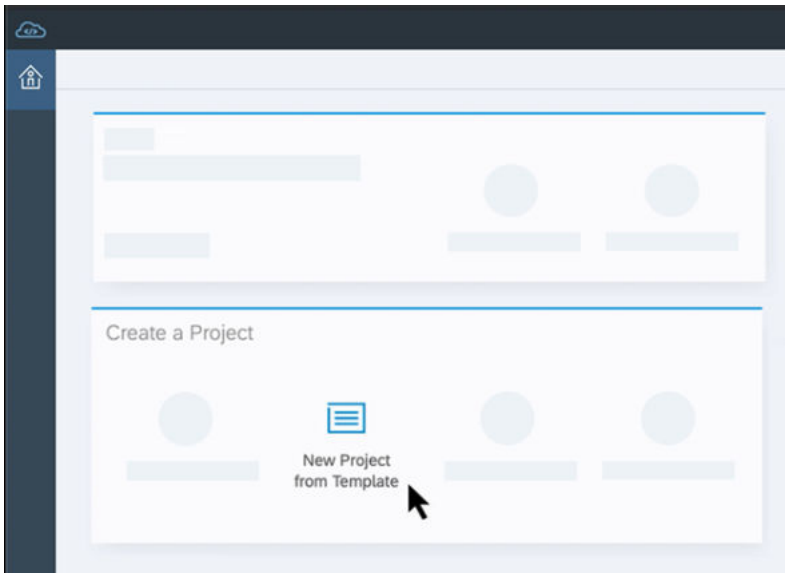
Asset Central Extensions is only supported for following applications:

- Equipment
- Models
- Locations
- Spare Parts
- Systems
- Failure Modes
- Functional Locations

Once you have activated the SDK, you can generate an asset central foundation application by creating a project in SAP Web IDE based on the SDK templates.

Procedure

1. In SAP Web IDE, create a new project by going to the home screen and choosing *New Project from Template*:

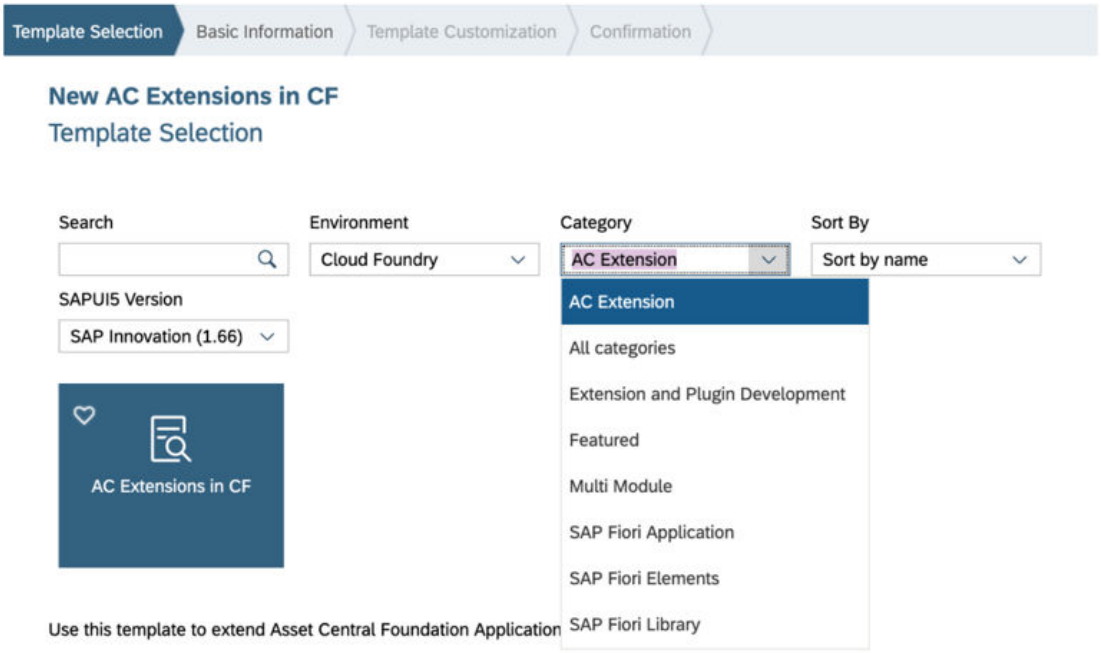


A new dialog box with the template selection as a first screen is displayed.

2. On the screen, fill in the following fields:

Environment	Select *Cloud Foundry*.
Category	Select *All categories* or *AC Extension*. **→ Remember** To see the *AC Extension*, you need to have created a destination for the SDK and have activated the SDK.

3. Select the template that you want to use as the technology for the asset central foundation application. You can choose: *AC Extensions in CF*



4. Choose *Next*.

   The screen for the basic information is displayed.

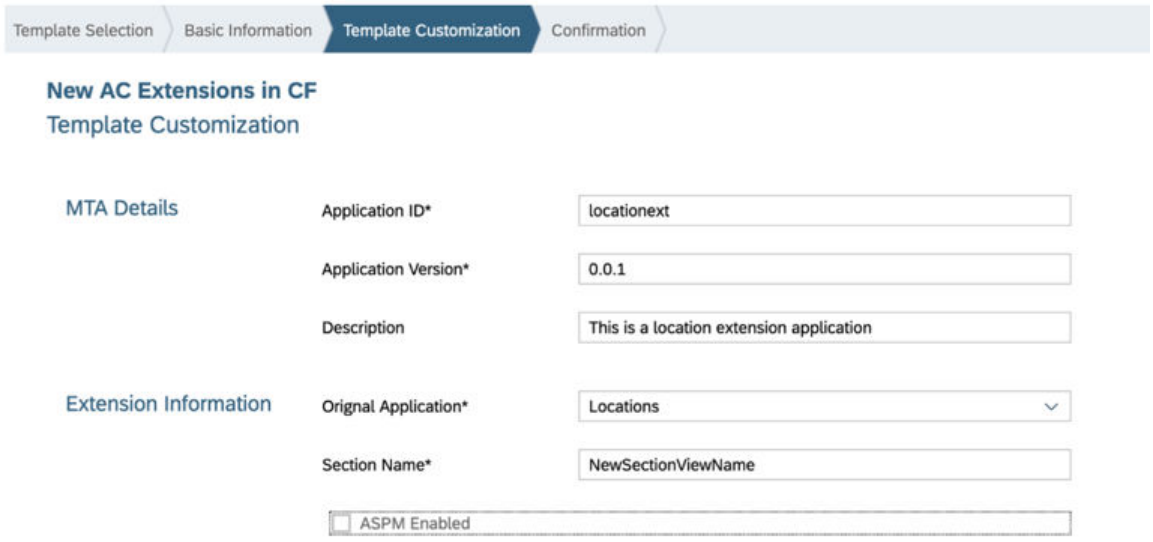
5. On the screen, enter a project name.
6. Choose *Next*.

   The screen for the template customization is displayed.

7. On the screen, fill in the following fields:

Application ID	Enter an ID for the application.
Application Version	Enter a version for the application.
Description	Enter a description to provide additional information about the application.
Original Application	Select the application that you want to extend from the drop-down.

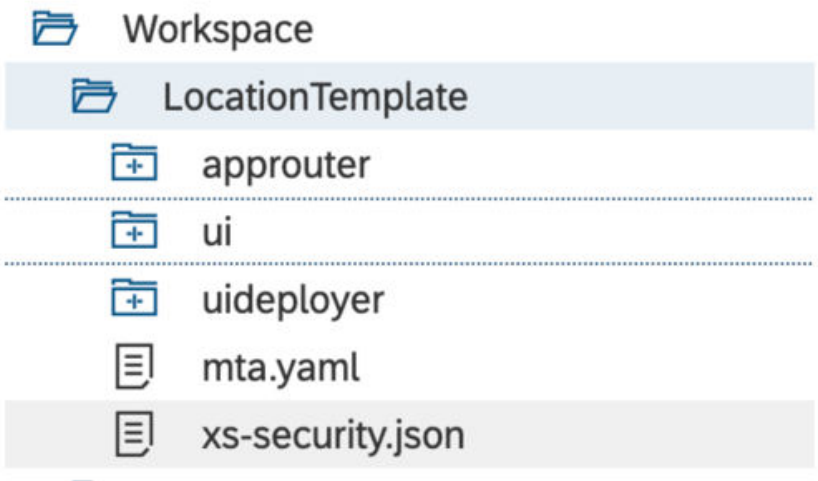
Section Name	Enter a view name for the new section.
ASPM Enabled	This is an optional field. Select the checkbox, only if you have subscribed to SAP Asset Strategy and Performance Management application.



8. Choose *Finish*.

Results

The application is generated and the new project is displayed in the *Files* tab on the *Development* screen. The following graphic shows an example of a generated *LocationTemplate* project:



Within the project, also multiple folders and files are generated that make the asset central foundation application fully working and allow you to create own variants of the generated application. For more

information about the generated folders and files, see Generated Folders and Files [page 12]. You can later delete and customize the generated folders and files according to your business scenario and what content you want to include in the application.

Related Information

Building and Deploying the Application [page 13]

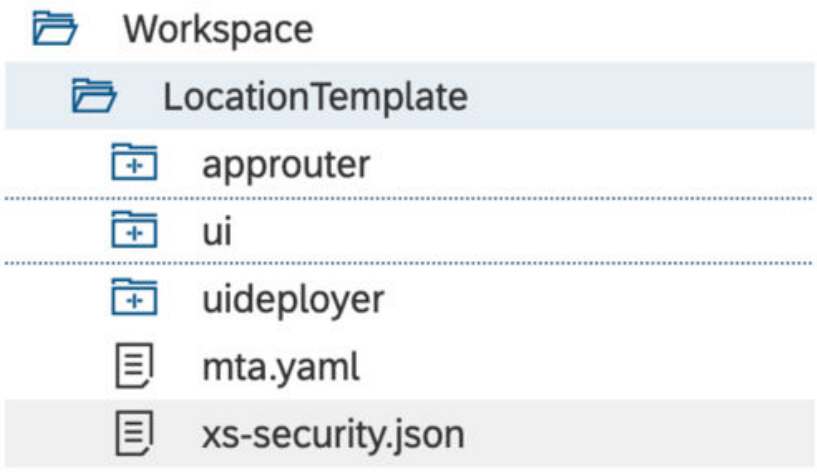
2.1.3.1 Generated Folders and Files

After you have generated an application, multiple folders and files are generated with the project. This topic provides you with an overview of the generated folders and files.

The completion of the project template wizard, generates the following three modules:

- App router module: This module is the single point-of-entry for locally developed application to the local SAP Fiori launchpad.
- UI module: this module contains the application source code.
- UI deployer: this module deploys all the source code from the UI module to the central HTML5 repository.

The following graphic shows an example of the *LocationTemplate* project with the generated folders and files:



Folders

Folder	Description
approuter	This folder contains metadata files that are used by SAP Web IDE and the multitarget application archive builder.
ui	This folder contains contains the application source code . It also contains i18n files, which include the name and description of the application.
ui-deployer	This subfolder contains a *package.json* file that deploys the content of the ui subfolder to the central HTML5 repository. The central HTML5 repository is the repository where all UIs are served from.

Files

File	Description
mta.yaml	This file contains information about the packaging, deployment, and the service-bindings of the modules. The modules are the api, db, ui and ui-deployer modules. Together they constitute the application. For more information, see Defining MTA Deployment Descriptors for Cloud Foundry.
xs-security.json	The XSUAA service for user authentication is configured using the security descriptor xs-security.json.

2.1.3.2 Building and Deploying the Application

This topic describes how to build and deploy the application.

Prerequisites

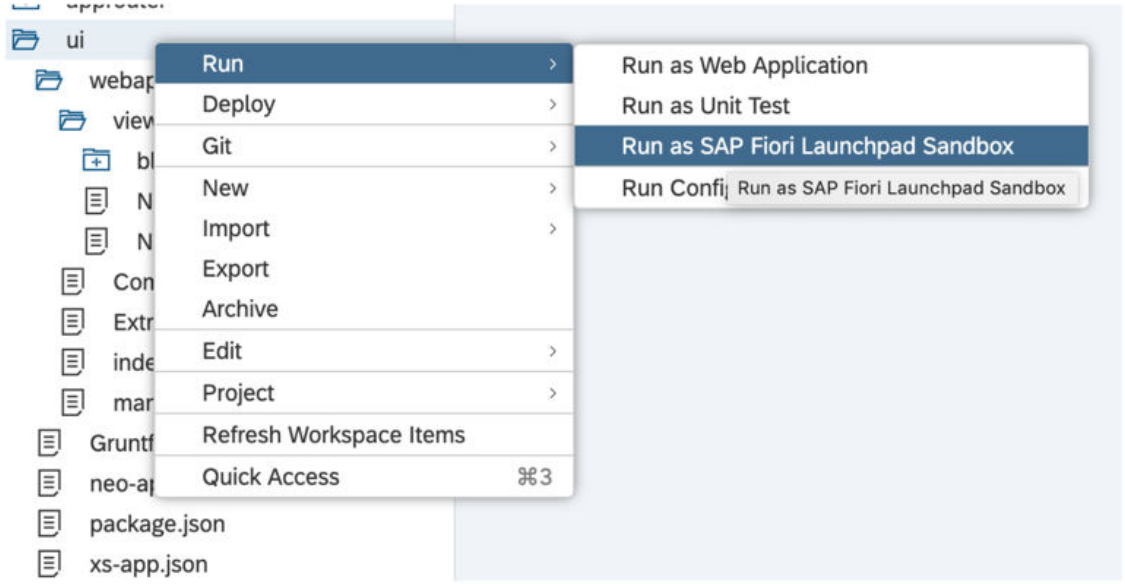
You have generated an set central foundation application. For more information, see Generating an Asset Central Foundation Extension Application [page 9].

Context

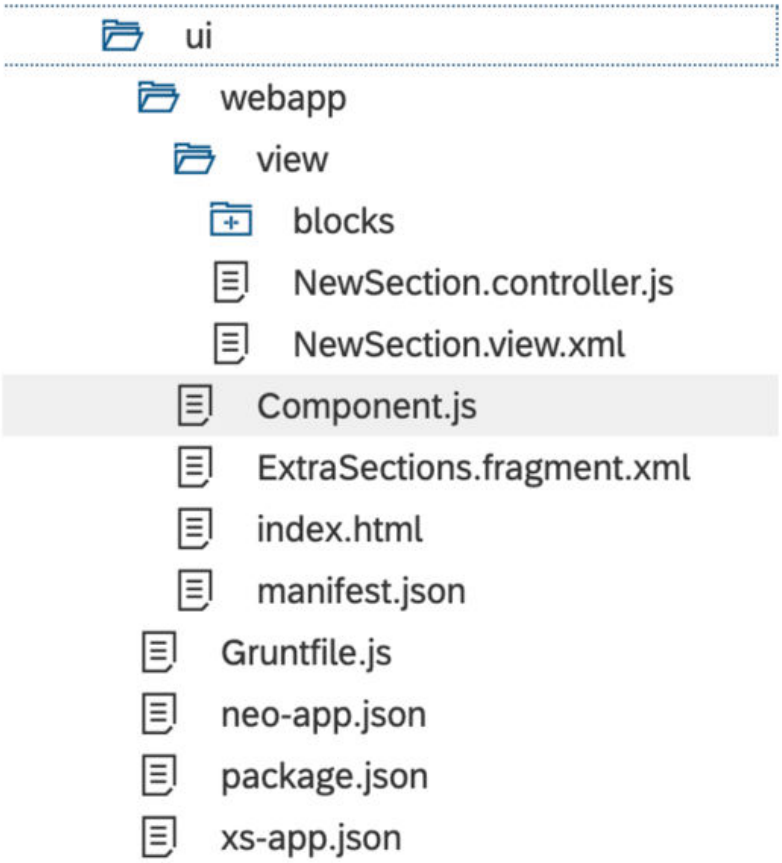
Once you have generated the application, you need to build it and deploy it to your space so that the application can be integrated into your own SAP Fiori launchpad.

Procedure

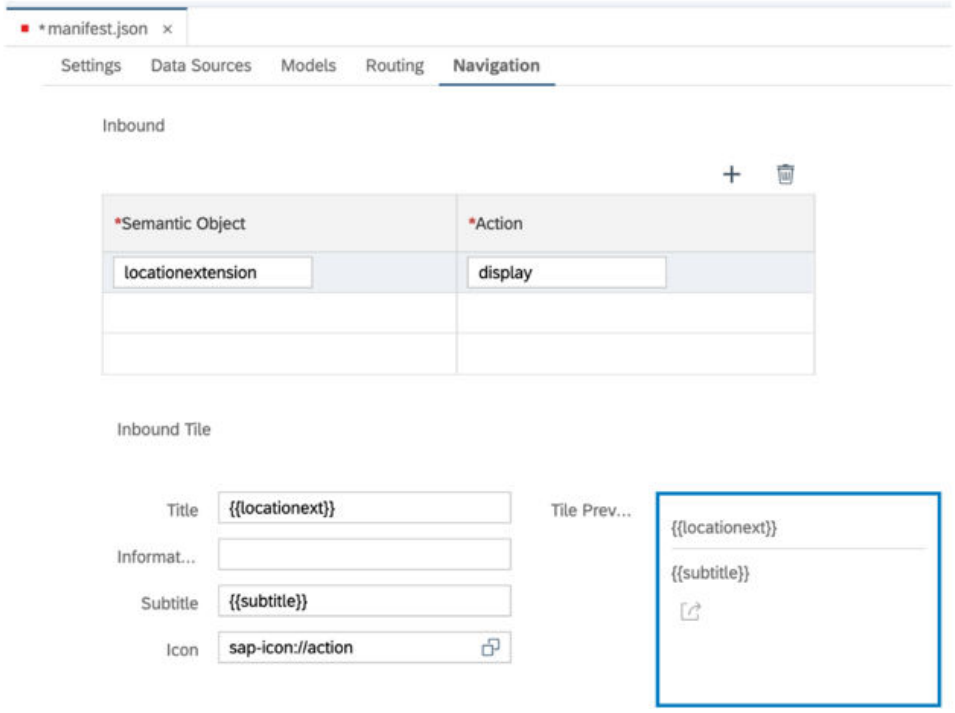
1. In the generated folder section, choose the **ui** folder.
2. In the context menu, choose *Run* > *Run as SAP Fiori Launchpad Sandbox*. The new section appears in the object page.



3. Add your implementation in the *NewSection* view and the controller is created.



4. Deploy to Cloud Foundry space after making changes to the UI application. The changes are as mentioned below:
    1. Open the manifest file.
    2. Switch to *Descriptor Editor*.
    3. Choose the *Navigation* tab.



4. Change sap-app version to 1.2.0.
5. Remove or comment the app router module from mta.yaml.

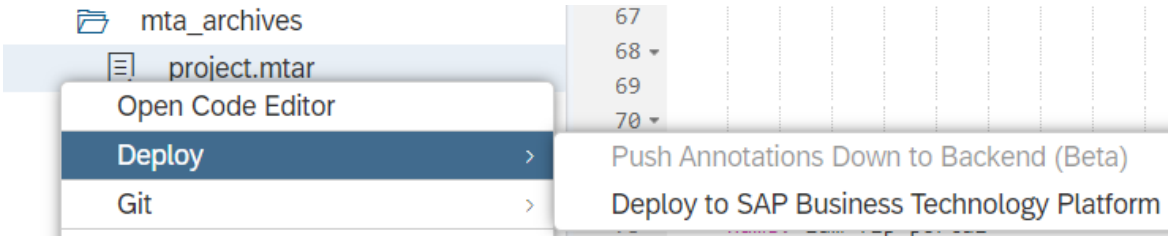
```
*mta.yaml ×
1  ID: mta_template3
2  _schema-version: '2.1'
3  parameters:
4    deploy_mode: html5-repo
5  version: 0.0.1
6  modules:
7    #- name: template3-approuter
8    #  type: approuter.nodejs
9    #  path: approuter
10   #  parameters:
11   #    disk-quota: 256M
12   #    memory: 256M
13   #  requires:
14   #    - name: repo_runtime
15   #    - name: flp_uaa
16   #    - name: asset-central-service-instance
17   #    - name: aspm-service-instance
```

6. Remove or comment all the services except the app-host service in the resources section of mta.yaml.

7. Comment the uncommented code and uncommon the commented code in component.js.

```
sap.ui.getCore().loadLibrary("sap.iot.ain.lib.reusable", "/comsapdsciamassetcentral.sapiotainlibreusable/");
sap.ui.getCore().loadLibrary("sap.iot.ain.lib.objects", "/comsapdsciamassetcentral.sapiotainlibobjects/");
sap.ui.getCore().loadLibrary("sap.dsc.aspm.lib.reusable", "/comsapdsciamaspm.sapdscaspmlibreusable/");
jQuery.sap.declare("sap.iot.ain.manageequipments.template3.Component");

// use the load function for getting the optimized preload file if present
sap.ui.component.load({
    name: "sap.iot.ain.manageequipments",
    url: "/comsapdsciamassetcentral.sapiotainmanageequipments/"
});

this.sap.iot.ain.manageequipments.Component.extend("sap.iot.ain.manageequipments.template3.Component", {
    metadata: {
        manifest: "json"
    }
});

/*

For CF Deployment

jQuery.sap.declare("sap.iot.ain.manageequipments.template3.Component");

// use the load function for getting the optimized preload file if present
sap.ui.component.load({
    name: "sap.iot.ain.manageequipments",
    // Use the below URL to run the extended application when SAP-delivered application is deployed on cloud
    // Remove the url parameter once your application is deployed to productive account
    url: jQuery.sap.getModulePath("sap.iot.ain.manageequipments.template3") + "/parent"
        // we use a URL relative to our own component
        // extension application is deployed with customer namespace
});

this.sap.iot.ain.manageequipments.Component.extend("sap.iot.ain.manageequipments.template3.Component", {
    metadata: {
        manifest: "json"
    }
});


*/
```

8. In the project context-menu, choose *Build*.
9. After the build is completed, the mtar file is generated in the *mta_archives* folder.
10. In the context menu of the mtar file, choose *Deploy* > *Deploy to SAP Business Technology Platform*.



Results

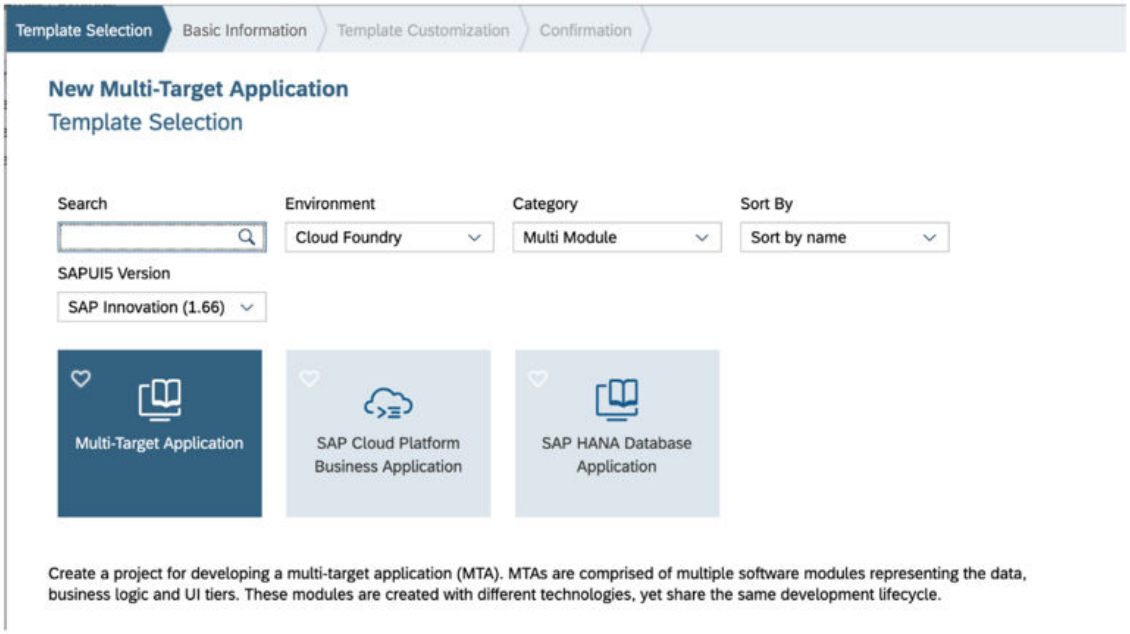
The application is built and deployed.

2.2 Extending Standalone Applications

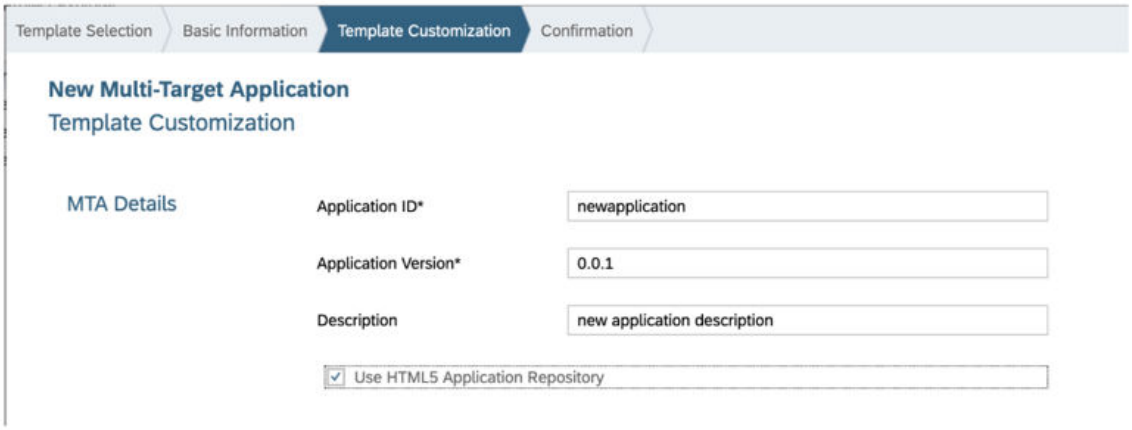
Context

Procedure

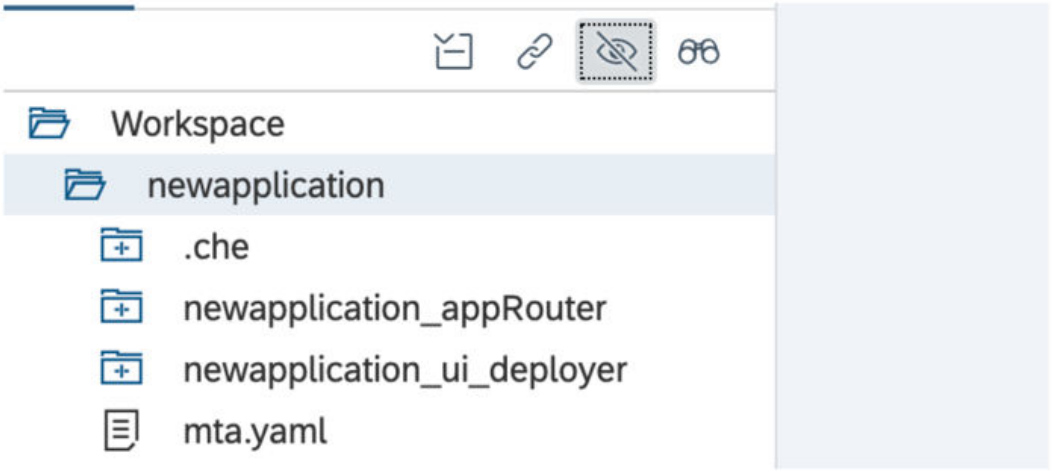
1. Create multi-target application using the steps provided below:
    1. Open SAP Web IDE.
    2. Navigate to the **(Development)** tab.
    3. Choose *File* > *New Project from Template*.
    4. In the wizard that appears, select *Cloud Foundry* environment and *Multi Module* category.
    5. Proceed with the *Multi-Target Application* template.



2. Customize asset central extension template. In the template customization step of the project creation wizard, enter the following details:
    - Application ID: enter an ID for the application
    - Application Version: enter a version for the application
    - Description: enter a logical description about the application
    - Use HTML5 Application Repository: Select the checkbox. This creates the UI Deployer module.

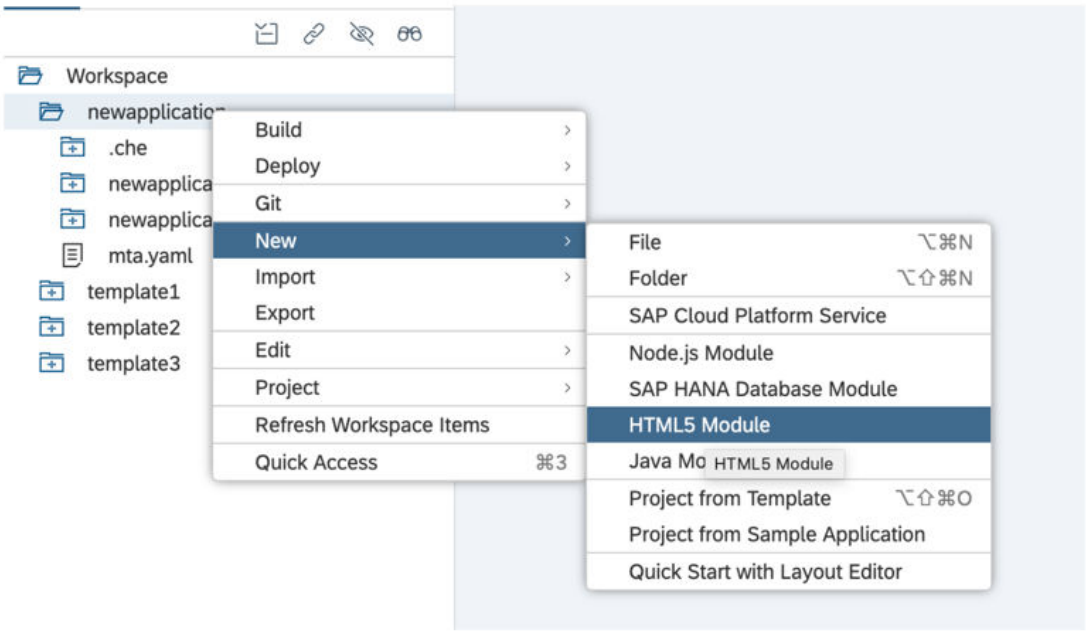


3. Create HTML5 module using the steps below:
    1. Choose the eye icon on the top of SAP Web IDE repository to see all the hidden modules. The app router and UI Deployer modules appear in the Workspace.



    2. In the context menu of the application, choose *New* > *HTML5 Module*.

> **i** Note
>
> You can use the SAPUI5 Application template to create a HTML5 module.



4. Add the required business service such as SAP Predictive Asset Insights, asset central foundation, SAP Asset Intelligence Network, or SAP Asset Strategy and Performance service to the xs-app.json file of the app router.

```
For SAP Predictive Maintenance and Service add the following details:
{
        "csrfProtection": false,
        "source": "^/pdms/(.*)$",
        "target": "/ain/$1",
        "service": "com.sap.dsc.iam.pdms",
        "endpoint": "pdms-backend",
        "authenticationType": "xsuaa"
    }
```

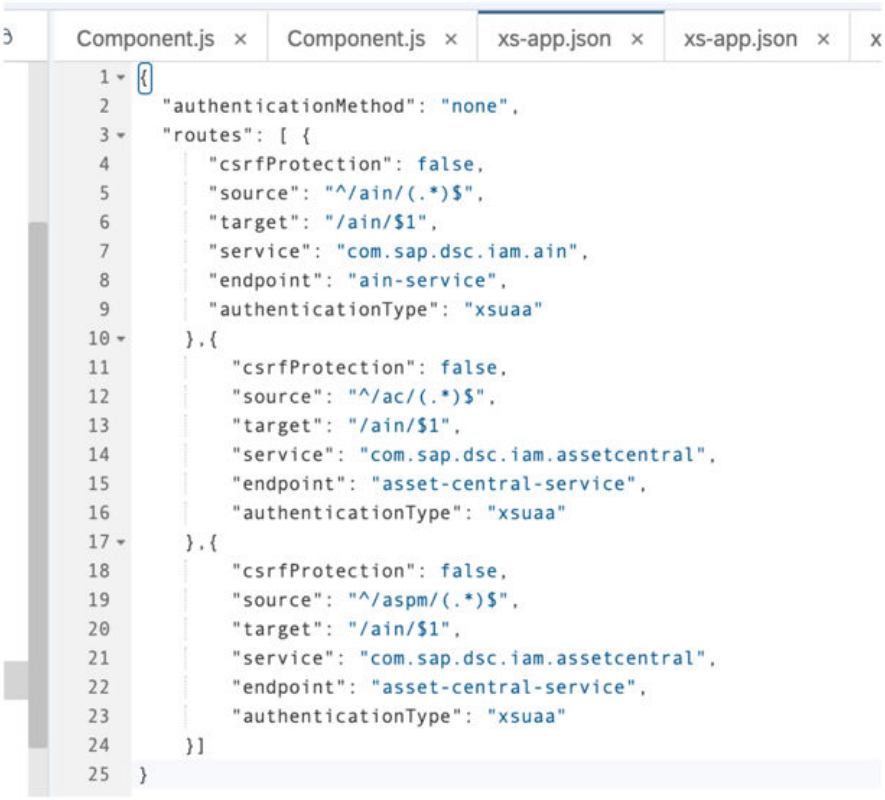
```
For Asset Central Service addd the following details:
{
        "csrfProtection": false,
        "source": "^/ac/(.*)$",
        "target": "/ain/$1",
        "service": "com.sap.dsc.iam.assetcentral",
        "endpoint": "asset-central-service",
        "authenticationType": "xsuaa"
    }
```

```
For SAP Asset Intelligence Network service add the following details:
{
      "csrfProtection": false,
      "source": "^/ain/(.*)$",
      "target": "/ain/$1",
      "service": "com.sap.dsc.iam.ain",
      "endpoint": "ain-service",
      "authenticationType": "xsuaa"
    }
```

```
For SAP Asset Strategy and Performance Management service add the following
details:
{
        "csrfProtection": false,
```

```
        "source": "^/aspm/(.*)$",
        "target": "/ain/$1",
        "service": "com.sap.dsc.iam.assetcentral",
        "endpoint": "asset-central-service",
        "authenticationType": "xsuaa"
    }
```

The required business service after additon appears as below:



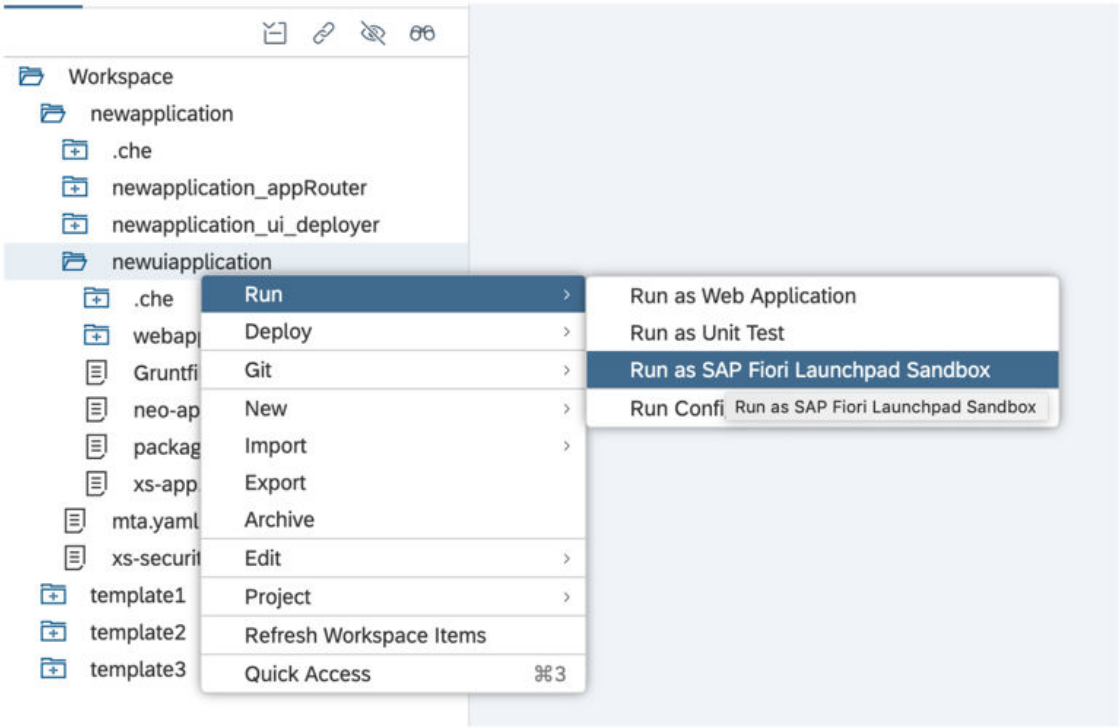
5. Bind the business service to the app router using the steps below:
    1. Open the mtad.yaml file.
    2. Add the required business service details under *resources*. The business service after addition appears as below:

```
resources:
  - name: asset-central-service-instance
    type: org.cloudfoundry.managed-service
    parameters:
      service: asset-central-poc
      service-plan: standard
  - name: aspm-service-instance
    type: org.cloudfoundry.managed-service
    parameters:
      service: aspm-poc
      service-plan: standard
  - name: ain-service-instance
    type: org.cloudfoundry.managed-service
    parameters:
      service: ain-poc
      service-plan: standard
```

3. Bind the business service to application router by adding it under *requires*. The business service after addition appears as below:

```
parameters:
  deploy_mode: html5-repo
version: 0.0.1
modules:
  - name: appRouter
    type: approuter.nodejs
    path: appRouter
    parameters:
      disk-quota: 256M
      memory: 256M
    requires:
      - name: asset-central-service-instance
      - name: ain-service-instance
      - name: aspm-service-instance
```

6. Running application on local SAP Fiori Launchpad.
    1. Choose the UI module.
    2. In the context menu, choose *Run* > *Run as SAP Fiori Launchpad Sandbox*. The new section appears in the object page.



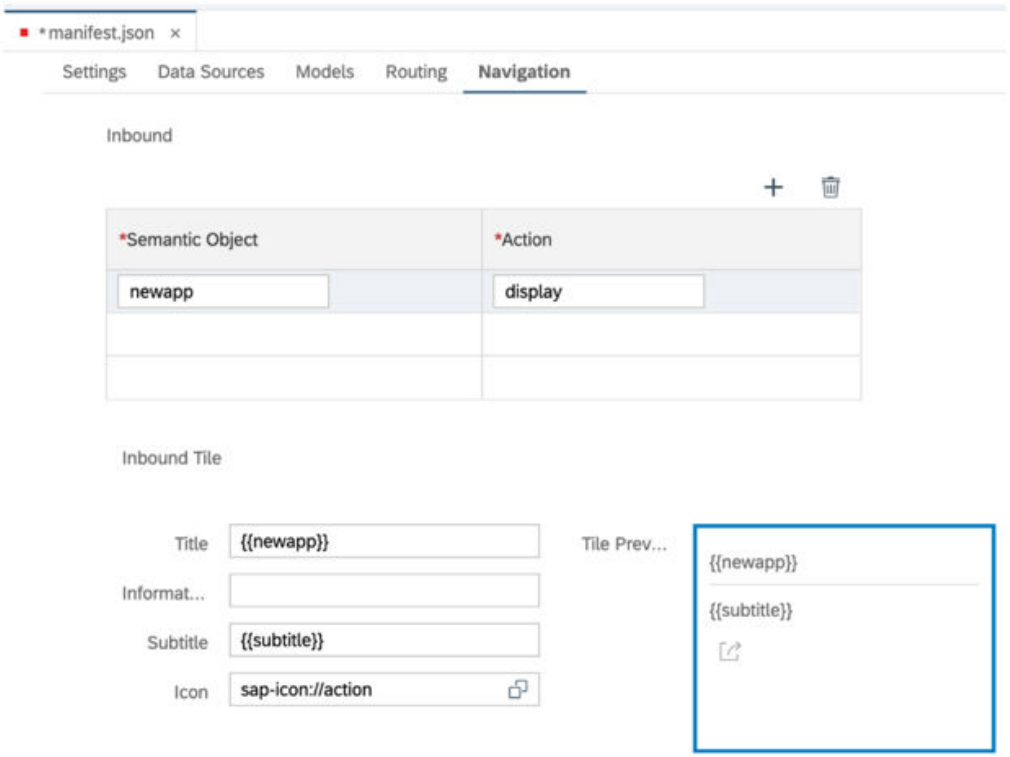
2.2.1 Building and Deploying the Application

Context

Deploy to Cloud Foundry space after changing the UI application. The changes are as mentioned below:

Procedure

1. Open the manifest file.
2. Switch to *Descriptor Editor*.
3. Choose the *Navigation* tab.



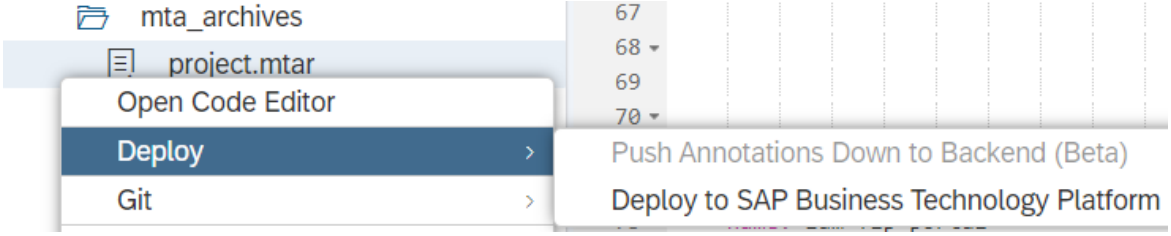
4. Change sap-app version to 1.2.0.
5. Remove or comment the app router module from mta.yaml.

```
    build-parameters:
      builder: grunt
resources:
  #- name: newapplication_html5_repo_runtime
  #  parameters:
  #    service-plan: app-runtime
  #    service: html5-apps-repo
  #  type: org.cloudfoundry.managed-service
  - name: newapplication_html5_repo_host
    parameters:
      service-plan: app-host
      service: html5-apps-repo
    type: org.cloudfoundry.managed-service
  #- name: uaa_newapplication
  #  parameters:
  #    path: ./xs-security.json
  #    service-plan: application
  #    service: xsuaa
  #  type: org.cloudfoundry.managed-service
  #- name: dest_newapplication
  #  parameters:
  #    service-plan: lite
  #    service: destination
  #  type: org.cloudfoundry.managed-service

```

6. Remove or comment all the services except the app-host service in the resources section of mta.yaml.

```
*mta.yaml ×
33        builder: grunt
34  resources:
35    - name: template3-dt
36      parameters:
37        service-plan: app-host
38        service: html5-apps-repo
39      type: org.cloudfoundry.managed-service
40    #- name: repo_runtime
41    #  parameters:
42    #    service-plan: app-runtime
43    #    service: html5-apps-repo
44    #  type: org.cloudfoundry.managed-service
45    #- name: flp_uaa
46    #  parameters:
47    #    path: ./xs-security.json
48    #    service-plan: application
49    #    service: xsuaa
50    #  type: org.cloudfoundry.managed-service
51    #- name: asset-central-service-instance
52    #  type: org.cloudfoundry.managed-service
53    #  parameters:
54    #    service: asset-central
55    #    service-plan: standard
56    #- name: aspm-service-instance
57    #  type: org.cloudfoundry.managed-service
58    #  parameters:
59    #    service: aspm
60    #    service-plan: standard
61
```

7. Comment the uncommented code and uncommon the commented code in component.js.
8. In the project context-menu, choose *Build*.
9. After the build is completed, the mtar file is generated in the *mta_archives* folder.
10. In the context menu of the mtar file, choose *Deploy* > *Deploy to SAP Business Technology Platform*.



2.3 Custom Fiori Launchpad with Asset Central UI Applications

2.3.1 Required and Optional Entitlements

In your global account and subaccount, multiple entitlements need to be assigned. In addition to the required entitlements, you can also assign additional entitlements depending for which subscription you want to build and deploy your own SAP Fiori launchpad. This topic provides you with an overview of all required and optional entitlements.

Required Entitlements

Services

Entitlement	Plan	Minimum Quota	Information on Requirement	Information on Pricing	More Information
Cloud Foundry Runtime	MEMORY	1 GB	The Cloud Foundry runtime service is required by the router of the SAP Fiori launchpad so that it can serve HTML5 content at runtime.	For more information about the prices for the services, see Your Estimate.	For more information about the services, see SAP Discovery Center.
Destination	lite	1 Unit	The destination service is required by the router of the SAP Fiori launchpad to find the relevant destination information to access the SAP Fiori launchpad from your Cloud Foundry application.		

Entitlement	Plan	Minimum Quota	Information on Requirement	Information on Pricing	More Information
HTML5 Application Repository Service	app-host	1-2 Units per application	The HTML5 application repository service is required to upload the UI code to the HTML5 repository from where the SAP Fiori launchpad takes it for displaying the UI. Depending on the application complexity like number of pages, lines of code and so on, the units can vary from 1-4.		
Cloud Portal Service	standard	1 Unit	The cloud portal service is required to deploy your own SAP Fiori launchpad to your space. You need to deploy your own launchpad to your space so that the extension modules including the UI and backend are integrated with the existing application and routes.		

Entitlement	Plan	Minimum Quota	Information on Requirement	Information on Pricing	More Information
UI5 Flexibility Service for Key Users	keyuser	1 Unit	The UI5 flexibility service for key users lets you provide UI adaptation capabilities for your UI5 applications on Cloud Foundry. Users of your applications can change the user interface of your applications in an upgrade-safe and modification-free way, without affecting any other customer.		
Network Spatial Service (SBN NSS Service)	Standard	1 Unit	This service impacts only the *Work Order* application. It displays Network Spatial map on the Work Order / Work Order Operation Object pages within the **Geometry** tab. For more information, see Network Spatial Service [page 30].		

Applications

Entitlement	Plan	Minimum Quota	Information on Requirement	Information on Pricing	More Information
Asset Central Foundation	standard	1 Unit	The applications are required because SAP Asset Intelligence Network is dependent on these applications in order to work correctly. The applications are also required to deploy the SAP Fiori launchpad in your space.	All applications are already included in your SAP Asset Intelligence Network standard license and do not need to be purchased.	Analytics, Processes, Master Data, and Administration
SAP Asset Intelligence Network	standard	1 Unit			SAP Asset Intelligence Network

2.3.1.1 Network Spatial Service

This service impacts only the *Work Order* application. It displays Network Spatial map on the work order or work order operation object pages within the *Geometry* tab.

If you do not wish to use the service, configure your Fiori launchpad **CommonDataModel json** with the following intent detail:

- semanticObject = ainworkorders
- action= displayWithoutNss

When this configuration is done, the *Geometry* section will not be visible on work order object page and operation object page.

If you want to include Network Spatial Service in your Fiori launchpad, you must update the following files:

- Add the below mentioned routes in `Fiori launchpad approuter xs-app.json`:

Sample Code

```
{
                "source": "^/nss/config(.*)$",
                "target": "$1",
                "service": "com.sap.bnc.nss.service",
                "endpoint": "coreservice",
                "authenticationType": "xsuaa"
        }, {
                "source": "^/nss/service(.*)$",
                "target": "$1",
                "service": "com.sap.bnc.nss.service",
```

```
                "endpoint": "coreservice",
                "csrfProtection": true,
                "authenticationType": "xsuaa"
        }, {
                "source": "^/odata/v2/(.*)$",
                "target": "/odata/v2/$1",
                "service": "com.sap.bnc.nss.service",
                "endpoint": "coreservice",
                "csrfProtection": true,
                "authenticationType": "xsuaa"
        }
```

- Add the Network Spatial Service instance resource to your `MTA.YML` file. This resource needs to added as a dependency to your Fiori launchpad approuter:

Sample Code

```
- name: nss-serviceinstance
    parameters:
      service-plan: standard
      service: bnc-nss
    type: org.cloudfoundry.managed-service
```

2.3.2 Building and Deploying Applications to Custom Fiori Launchpad

Prerequisites

- You have installed the Cloud Foundry command line tool with the multi app plugin (cf install-plugin multiapps).
- The `Space Developer` role is assigned to your user for the deployment in your target space.

Context

This topic contains steps to setup custom Fiori launchpad and add extension to the custom Fiori launchpad.

Procedure

1. Extract contents of the attached archive. The archive has pre-delivered SAP Fiori launchpad content based on the product (PAI or AIN or ASPM or DVH). The table below lists subscriptions with its corresponding links to download the archive:

Subscription	Link to download archive
PAI	https://help.sap.com/doc/7b091caac9034fb1adf895c44c7dd549/2210/en-US
AIN	https://help.sap.com/doc/b786a33c661f4db88231276fb84fcab8/2210/en-US
ASPM	https://help.sap.com/doc/3a1a3d82078e487fbc6dce-dea5ea4598/2210/en-US
AIN and ASPM	https://help.sap.com/doc/7d9830a7b74b4975a3c71080a2fdb179/2210/en-US
AIN and PAI	https://help.sap.com/doc/d619f1f1c05f401ebe86ac72a586d90a/2210/en-US
ASPM and PAI	https://help.sap.com/doc/7169830e7eb7443eb6a40609c7145fbb/2210/en-US
AIN, ASPM, and PAI	https://help.sap.com/doc/93b788dd041740f29a6674c0963622e3/2210/en-US
AIN, DVH, and PAI	https://help.sap.com/doc/a9270ea0fb6f498b8a95647deeac6107/2210/en-US
AIN and DVH	https://help.sap.com/doc/112a16f8721a492f838ede40caf8914e/2210/en-US
AIN, ASPM, and DVH	https://help.sap.com/doc/c7cb254d7aca4d9988f13843114fd6c4/2210/en-US
AIN, ASPM, PAI, and DVH	https://help.sap.com/doc/d7e5bee12def4267a78c0ed6800f9e9c/2210/en-US

2. Open *CommonDataModel.json* in the *portal-site* folder. The structure of this file is explained in the README file available within the archive.
3. Add the ID property of the new extension application in the required group and catalog json objects within the structure as for other apps.

   For Catalog

   - “appId” property is the “id” property of the application from the *manifest.json* file for the extension application
   - “vizId” is the semantic object action from the tile configuration maintained in the *manifest.json* of the extension application.

   ```
   {
       "appId": "sap.iot.ain.manageequipments.extensionapplication",
       "vizId": "ainequipmentextension-display"
   },
   ```

   For Group

- "id" property is any unique identifier text.
- "appId" is the "id" of application property from the manifest.json file of the application.
- "vizId" is the semantic object action from the tile configuration maintained in the manifest.json of the application.

```
{
    "id": "someRandomExtension-id",
    "appId": "sap.iot.ain.manageequipments.extensionapplication",
    "vizId": "ainequipmentextension-display"
},
```

4. In the *mta.yaml* file of the extension application, find the name of the *html5-apps-repo* service instance.

```
resources:
 - name: extension-dt
   parameters:
     service: html5-apps-repo
     service-plan: app-host
   type: org.cloudfoundry.existing-service
```

5. Bind this service instance to the Fiori launchpad module under the *requires* section in the *mta.yaml* file.

```
 - name: ain-flp
   type: com.sap.portal.site-content
   path: portal
   parameters:
     stack: cflinuxfs3
     deploy_mode: html5-repo
     memory: 128M
     buildpack: 'https://github.com/cloudfoundry/no
   requires:
     - name: extension-dt
```

6. Maintain a corresponding entry under the *resources* section for this service similar to other bound services.

```
resources:
 - name: extension-dt
   parameters:
     service: html5-apps-repo
     service-plan: app-host
   type: org.cloudfoundry.existing-service
```

7. Build and deploy the custom Fiori launchpad in your development space after all the changes have been made.

2.4 Working with Destinations

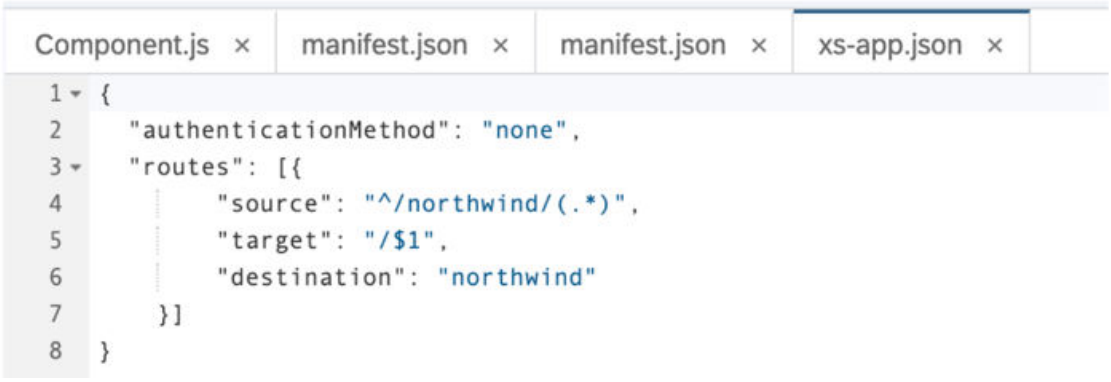
Context

Working with Destinations for Local Deployment

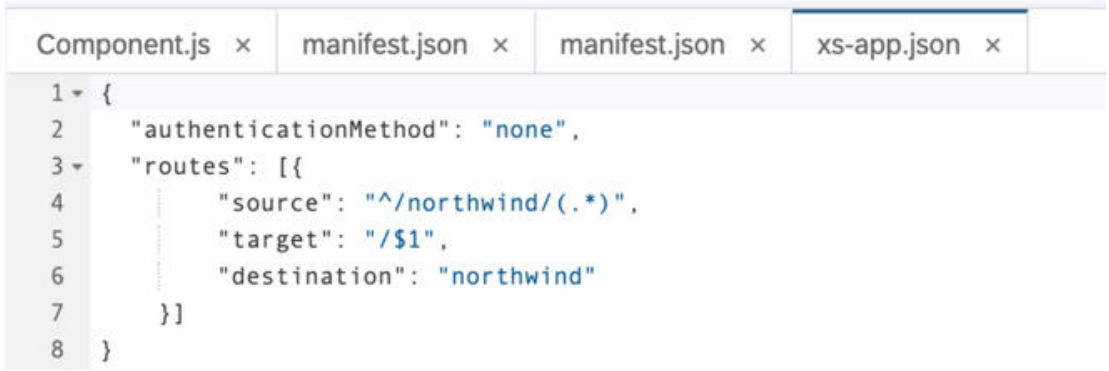
1. Add the destination service in the mta.yaml file of the new extension application in the ‘resources’ section. If you want to re-use an existing destination service, ensure that you keep the ‘name’ and ‘service-name’ properties value the same as in the instance that has already been created. The same applies to the other services.

```
resources:
  - name: destination-service
    parameters:
      service-plan: lite
      service: destination
      service-name: destination-service
    type: org.cloudfoundry.managed-service
```

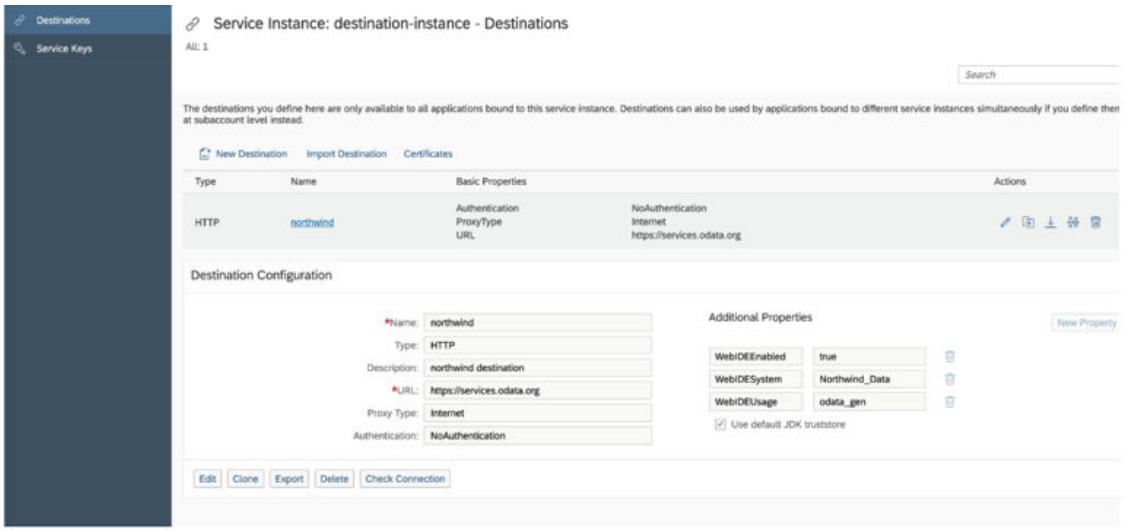
2. Add the destination in the required section for the app router module of your new application.



3. Configure the routes in the xs-app.json file for the app router.



4. Navigate to the newly-created destination service in your Cloud Foundry space and maintain the relevant destination.



Working with Destinations for Cloud Foundry Deployment

1. Add the destination service in the mta.yaml file of the ain-flp project in the ‘resources’ section. If you want to re-use an existing destination service, ensure that you keep the ‘name’ and ‘service-name’ properties value the same as in the instance that has already been created. The same applies to the other services.

```
resources:
  - name: destination-service
    parameters:
      service-plan: lite
      service: destination
      service-name: destination-service
    type: org.cloudfoundry.managed-service
```

2. Add the destination in the required section for the app router module.

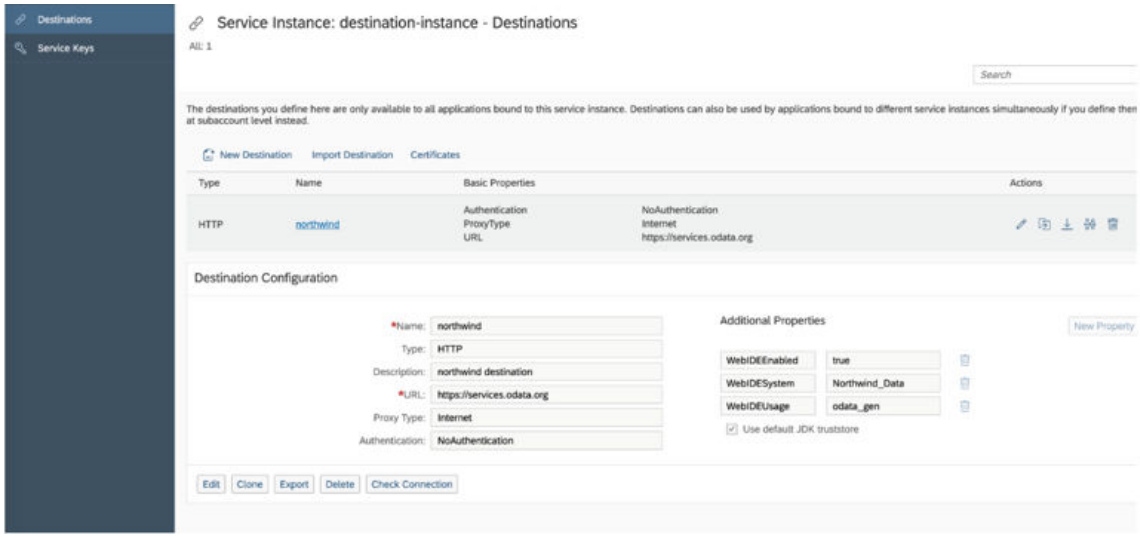
```
ID: ain-flp
_schema-version: '2.1'
description: Fiori Launchpad content for Asset Intelligence Network
version: 0.0.1
modules:
 - name: ain-flp-approuter
   type: javascript.nodejs
   path: approuter
   parameters:
     disk-quota: 256M
     memory: 256M
     keep-existing-routes: true
     stack: cflinuxfs3
     buildpack: 'https://github.com/cloudfoundry/nodejs-buildpack/releases/download
   requires:
     - name: destination-service
```

3. Configure the routes in the xs-app.json file for the app router in the ain-flp project.

Component.js × | manifest.json × | manifest.json × | xs-app.json ×

```
{
  "authenticationMethod": "none",
  "routes": [{
      "source": "^/northwind/(.*)",
      "target": "/$1",
      "destination": "northwind"
    }]
}
```

4. Navigate to the newly created destination service in your Cloud Foundry space and maintain the relevant destination.



3 Customizing applications using Adapt UI

With UI adaptation enabled, you can customize the applications as suitable for your organization and processes.

If Adapt UI is enabled for an application, you can customize the applications to rename field labels, rearrange the order of the fields, remove fields that aren't applicable to your organization.

Currently, the following applications in SAP Asset Intelligence Network are enabled for using Adapt UI:

- Work Orders
- Notifications

Important Disclaimers and Legal Information

Hyperlinks

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

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

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

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

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

© 2022 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.

THE BEST RUN SAP