

Step-by-step instructions to create a Custom Fiori Launchpad Plugin





TABLE OF CONTENTS

INTRODUCTION	3
SYSTEM PREREQUISITES.....	3
TARGET GROUP	3
BUILDING A CUSTOM UI PLUGIN	4



INTRODUCTION

By executing the steps described in this document you will be able to understand the main procedures required to create a custom Fiori Launchpad Plugin.

This document is created as a step-by-step guide to demonstrate how to quickly create a Fiori Launchpad Plugin in S/4HANA On-Premise. By reading this document you will be able to understand the main activities required to activate App Finder and some of the common errors and solutions.

SYSTEM PREREQUISITES

- SAP S/4HANA 1809 On-Premise with Embedded Fiori deployment
- Technical Activation of Fiori Launchpad has been concluded
- Fiori Administrator Roles have been assigned to your user
- Developer user must possess an SAP Cloud Platform account with WebIDE Full-Stack.
- SAP Cloud Connector has been deployed and correctly configured in your system landscape either as a central instance or in each of the developer's desktops.
- Destinations to the required backend system have been setup correctly.
- Developers have all the required authorizations to deploy an UI object to the ABAP backend system
- Developers can create Workbench Transport Orders and Packages in the ABAP backend system

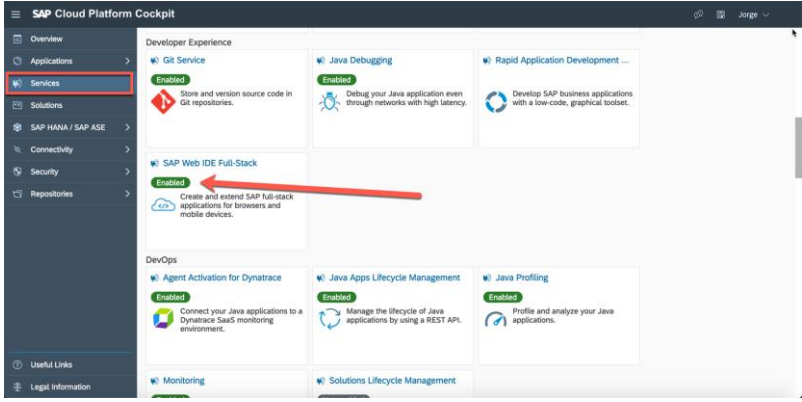
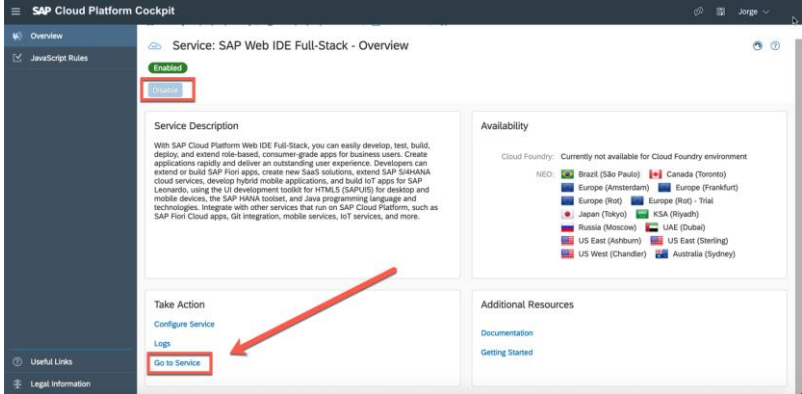
TARGET GROUP

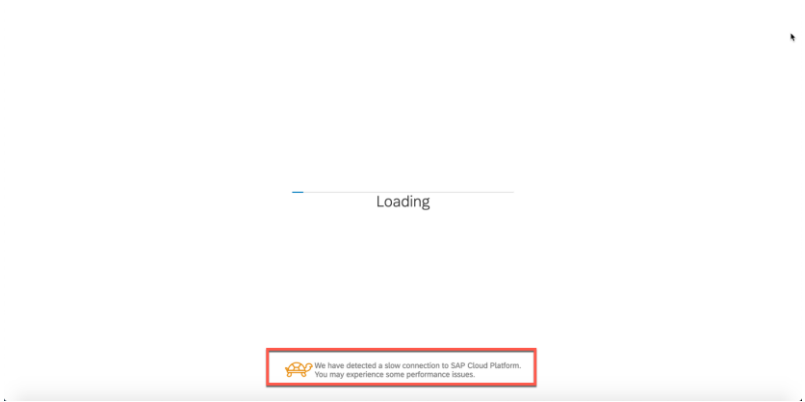
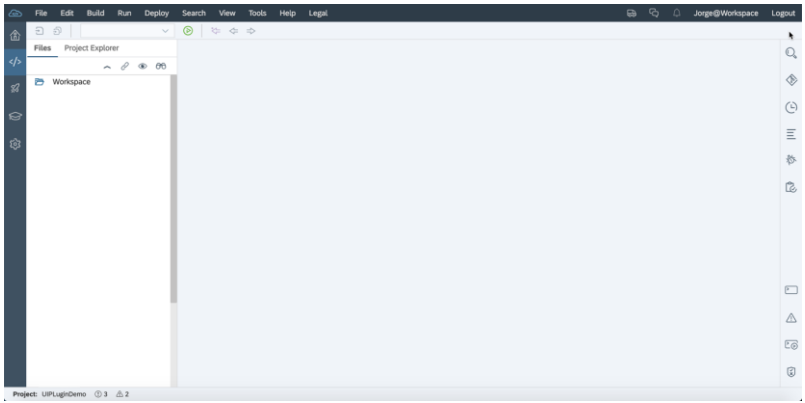
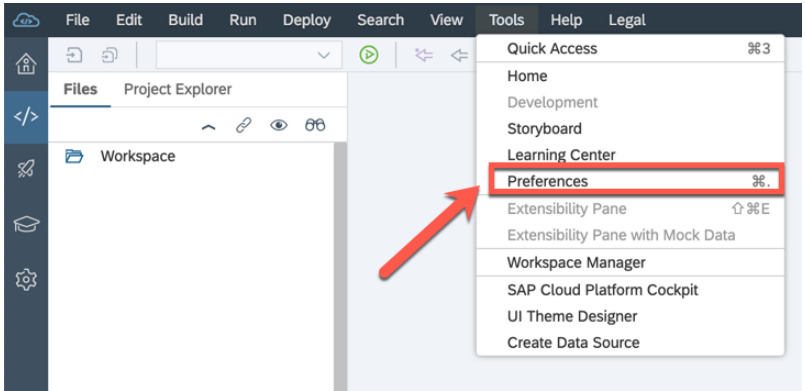
System Administrators
SAP Fiori Developers

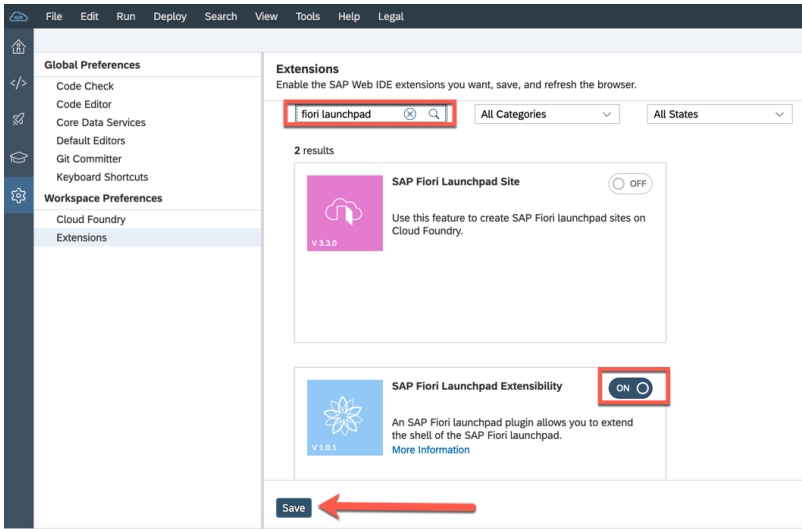
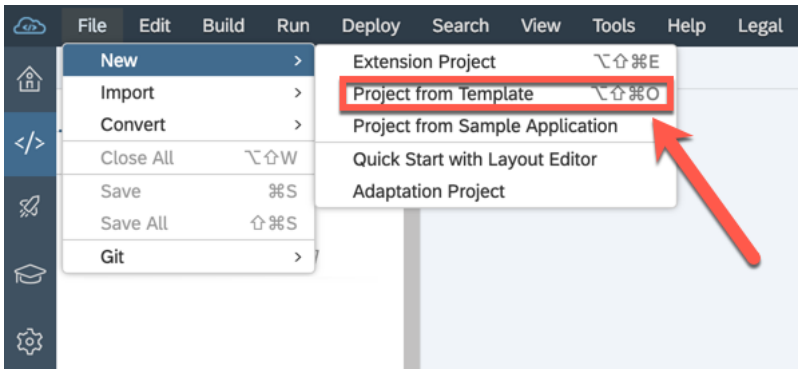
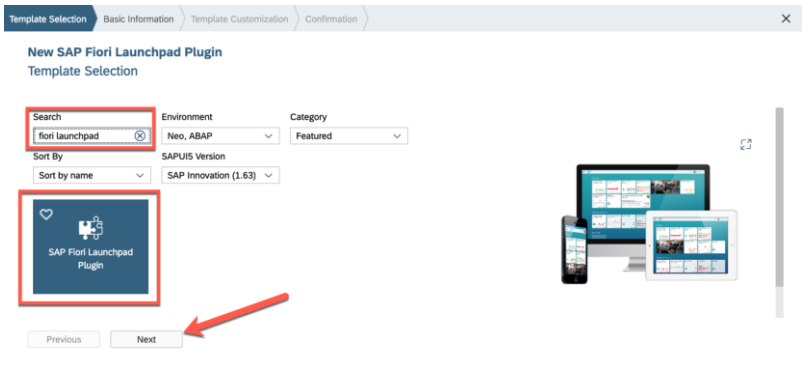
Estimated Execution Time: 45 minutes

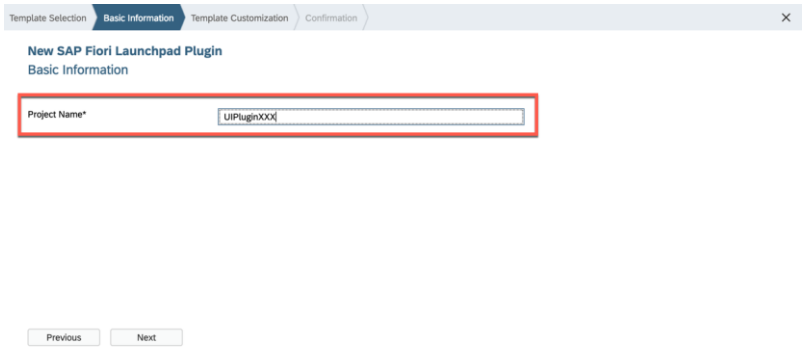
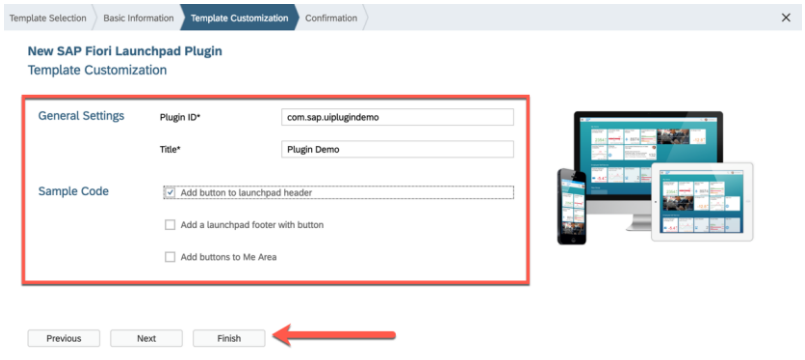
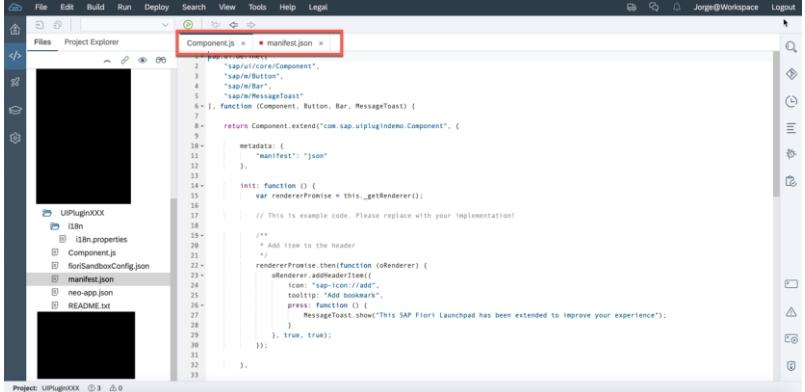
BUILDING A CUSTOM UI PLUGIN

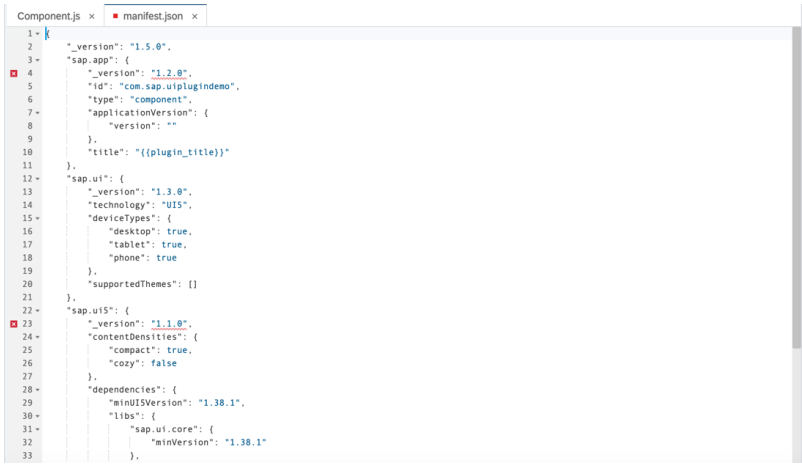
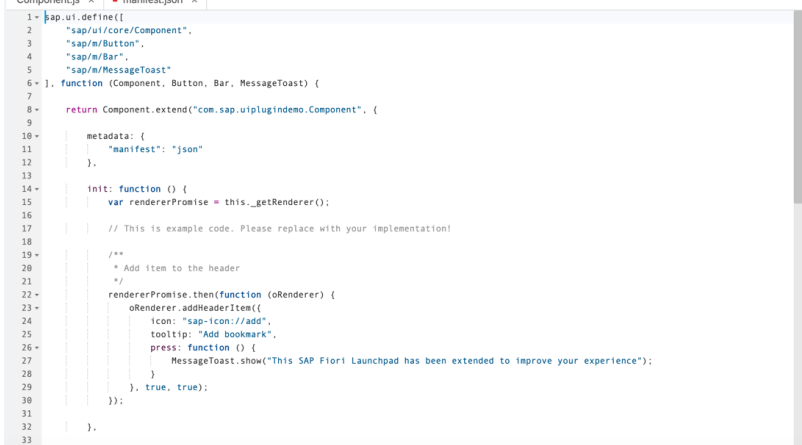
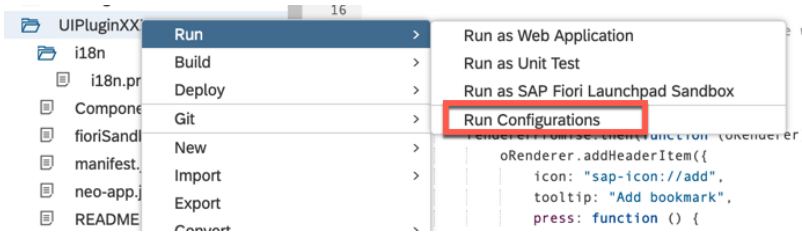
In the following section you will create a custom UI Plugin and deploy it to your Fiori Launchpad. The main objective for this UI Plugin is to load the current client number and User name in the Fiori Launchpad Header Text Area.

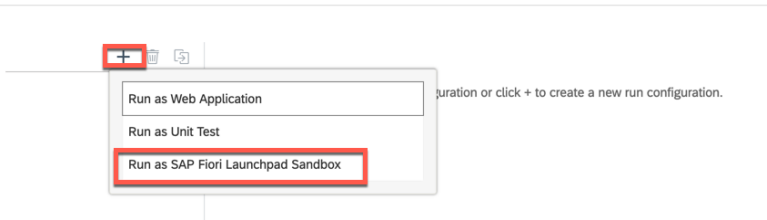
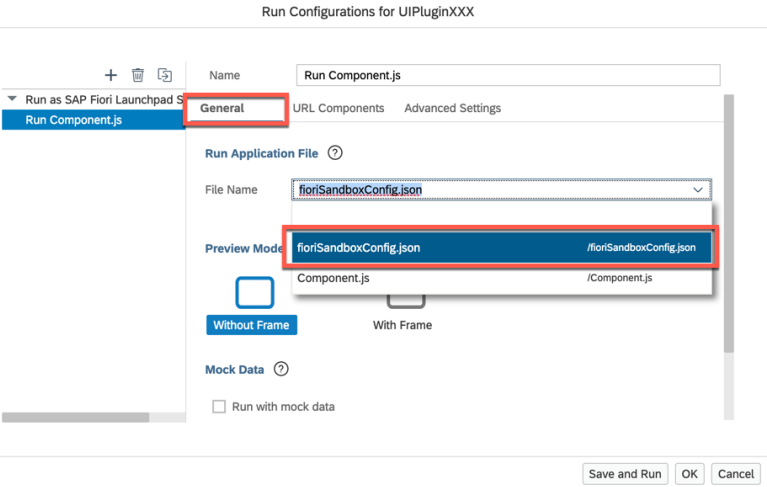
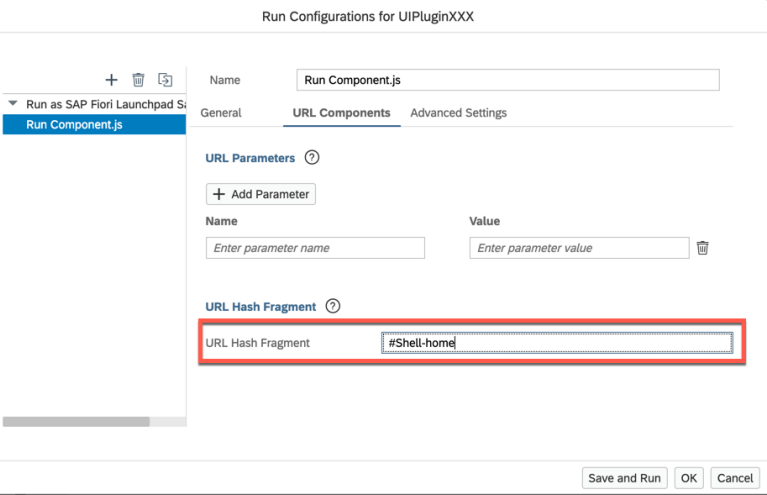
Steps / Comments	Screenshot
<ol style="list-style-type: none">1. Login to your SCP account and in the Services section, ensure SAP WebIDE Full-Stack is enabled.2. Click on the tile to gain access to the configuration options of this service	
<ol style="list-style-type: none">3. If you wish to Enable / Disable the service, click on the Enable / Disable button (the text changes depending on the status of your service)4. With the service in status Enabled, click on the Go to Service link to launch WebIDE.	

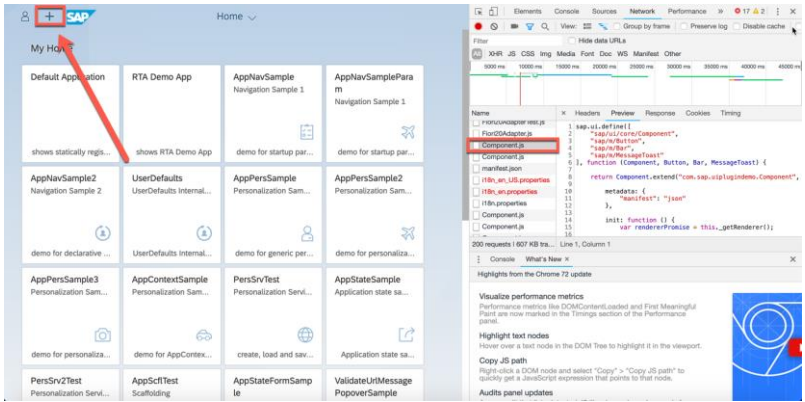

Steps / Comments	Screenshot
<p>5. WebIDE will be launched on a separate tab in your browser.</p> <p><i>**Wait for the application to finish loading. If your network connection is slow, you will notice a discrete notification stating possible performance issues</i></p>	
<p>6. Once the tool finishes loading, you will notice several work areas you will be using throughout the exercise</p>	
<p>7. To start building the new UI Plugin, you must first ensure that the required templates are available.</p> <p>8. In the toolbar, navigate to Tools → Preferences</p>	



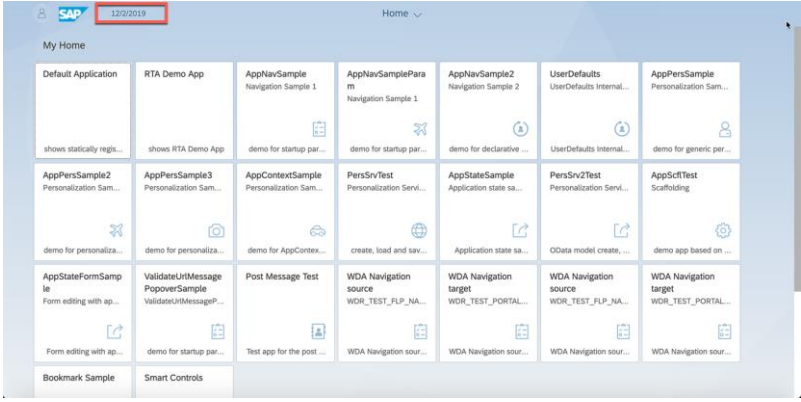
Steps / Comments	Screenshot
<p>9. Navigate to the Extensions section and search for the SAP Fiori Launchpad Extensibility extension.</p> <p>10. Make sure this extension is switched on.</p> <p>11. If not switch it on and press the Save button. Activating this extension will require you to reload the tool.</p>	
<p>12. With the SAP Fiori Launchpad Extensibility active you can now create a project out of a standard template.</p> <p>13. In the toolbar, navigate to File → New → Project from Template</p>	
<p>14. Search for the term “Fiori launchpad” and you will observe the SAP Fiori Launchpad plugin template.</p> <p>15. Select this template and click Next</p>	



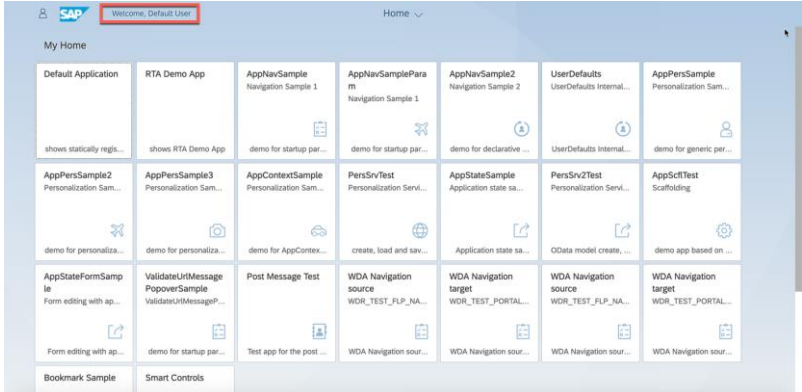
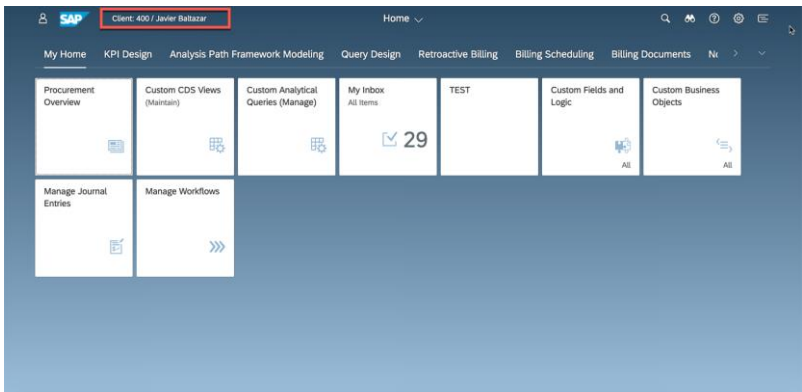
Steps / Comments	Screenshot
<p>16. Enter the name of your plugin project.</p> <p>17. For the purpose of this exercise we will use the name <i>UIPluginXXX</i> but you can define any project name.</p>	
<p>18. Fill in the details of your project. For this exercise we will use the following information:</p> <p>Plugin ID: <i>com.sap.uiplugindemo</i> Title: <i>Plugin Demo</i></p> <p>19. Make sure you select the checkbox to "add button to launchpad header" as this will add sample code we will take advantage from.</p> <p>20. Click finish once you are done with the settings.</p> <p>**Please note that you can define any Plugin ID however, this ID must remain unique in your system as it is used for runtime resolution.</p>	
<p>21. The wizard will now create your project structure with all the relevant code for the implementation.</p> <p>22. You can try exploring the contents of each file. Most importantly the content of two files which we will use for our custom plugin:</p> <p>Manifest.json Component.js</p>	

Steps / Comments	Screenshot
<p>23. In the Manifest.json file you will find the configuration parameters for the plugin. These are required by the Fiori Launchpad to be able to solve target mappings at runtime and identify how and when to load the UI Plugin.</p> <p>24. For the purpose of this exercise we will not modify any of the values in this file.</p>	 <pre> 1- { 2- "_version": "1.5.0", 3- "sap.app": { 4- "_version": "1.2.0", 5- "id": "com.sap.uiplugindemo", 6- "type": "component", 7- "applicationVersion": { 8- "version": "" 9- }, 10- "title": "({{plugin_title}})" 11- }, 12- "sap.ui": { 13- "_version": "1.3.0", 14- "technology": "UIS", 15- "deviceTypes": { 16- "desktop": true, 17- "tablet": true, 18- "phone": true 19- }, 20- "supportedThemes": [] 21- }, 22- "sap.ui5": { 23- "_version": "1.1.0", 24- "contentDensities": { 25- "compact": true, 26- "cozy": false 27- }, 28- "dependencies": { 29- "minUI5Version": "1.38.1", 30- "libs": { 31- "sap.ui.core": { 32- "minVersion": "1.38.1" 33- } 34- } 35- } 36- } 37- } </pre>
<p>25. In the Component.js file you will find the definition in Javascript of the methods that are used by this UI Plugin.</p> <p>26. UI Plugins follow the same design principles as Fiori Apps but they are only defined by the Component.js file which contains all the code to define the behavior of the Plugin.</p> <p>27. In this file you will currently find sample code to add a button to the Fiori Launchpad and display a Message Toast.</p>	 <pre> 1- sap.ui.define([2- "sap/ui/core/Component", 3- "sap/m/Button", 4- "sap/m/Bar", 5- "sap/m/MessageToast" 6-], function (Component, Button, Bar, MessageToast) { 7- 8- return Component.extend("com.sap.uiplugindemo.Component", { 9- 10- metadata: { 11- "manifest": "json" 12- }, 13- 14- init: function () { 15- var rendererPromise = this._getRenderer(); 16- 17- // This is example code. Please replace with your implementation! 18- 19- /** 20- * Add item to the header 21- */ 22- rendererPromise.then(function (oRenderer) { 23- oRenderer.addHeaderItem({ 24- icon: "sap-icon://add", 25- tooltip: "Add bookmark", 26- press: function () { 27- MessageToast.show("This SAP Fiori Launchpad has been extended to improve your experience"); 28- }, true, true); 29- }); 30- }); 31- }, 32- 33- _getRenderer: function () { 34- // ... 35- } 36- }); 37- 38- }); </pre>
<p>28. For starters, we will test the generated code in the Fiori Launchpad Sandbox of WebIDE to guarantee it is working.</p> <p>29. To do this you must first setup the runtime configurations of your project.</p> <p>30. Right click on your project folder name and navigate to Run → Run Configurations</p>	

Steps / Comments	Screenshot
31. Your configurations should be empty. Click on Add→Run as SAP Fiori Launchpad Sandbox	 The screenshot shows the 'Run Configurations for UIPluginXXX' dialog. On the left, there is a list of run configurations. The 'Run as SAP Fiori Launchpad Sandbox' option is highlighted with a red box. On the right, there is a description of the configuration.
32. In the General tab, change the Run Application File Name to fioriSandboxConfig.json 33. Leave all other values in this section as is.	 The screenshot shows the 'Run Configurations for UIPluginXXX' dialog, General tab. The 'Run Application File Name' is set to 'fioriSandboxConfig.json'. The 'Preview Mode' is set to 'Without Frame'. The 'Mock Data' checkbox is unchecked.
34. Switch to the URL Components tab and in the URL Hash Fragment , enter the values: #Shell-home 35. Click on the Save and Run button.	 The screenshot shows the 'Run Configurations for UIPluginXXX' dialog, URL Components tab. The 'URL Hash Fragment' is set to '#Shell-home'. The 'Save and Run' button is highlighted with a red box.

Steps / Comments	Screenshot
<p>36. A new browser tab will be opened, and you will now see the Fiori Launchpad Sandbox with an additional icon in the upper left side corner.</p> <p>37. Use the browsers developer tools to identify at what time your UI Plugin is loaded, this would be useful to identify which objects are available at the time your Plugin is loaded in case you want to define complex behaviors.</p>	
<p>38. You have now tested your code and can now start adding custom procedures. To do this we recommend using the generated <u>getRenderer function</u>.</p> <p>39. This is standard code which ensures the Plugin is loaded in a reliable way, independent from the initialization time of the plugin. Do not remove this function as we will reuse it for exercise purpose.</p>	
<p>40. In the init function add the following variables to obtain the current date and change the rendererPromise call to add text to the Fiori Launchpad Header Title.</p>	<pre>//Get Date var today = new Date(); var dd = today.getDate(); var mm = today.getMonth(); var yyyy = today.getFullYear(); var text1 = dd.toString() + "/" + mm.toString() + "/" + yyyy.toString(); /** * Add item to the header */ rendererPromise.then(function (oRenderer) { oRenderer.setHeaderTitle(text1); });</pre>

Steps / Comments	Screenshot
<p>41. Your code should look like in the image.</p> <p>42. Once you have made the adjustments, Save your changes with the save button  and test the plugin in the Fiori Launchpad Sandbox</p>	 <pre> 14 init: function () { 15 var rendererPromise = this._getRenderer(); 16 17 // This is example code. Please replace with your implementation! 18 19 //Get Date 20 var today = new Date(); 21 var dd = today.getDate(); 22 var mm = today.getMonth(); 23 var yyyy = today.getFullYear(); 24 var text1 = dd.toString() + "/" + mm.toString() + "/" + yyyy.toString(); 25 /** 26 * Add item to the header 27 */ 28 rendererPromise.then(function (oRenderer) { 29 oRenderer.setHeaderTitle(text1); 30 }); 31 }, 32 33 }, </pre>
<p>43. You will now see the current date in the Fiori Launchpad Header Title.</p>	
<p>44. To continue with this exercise we will now obtain the user's name and display it in the Fiori Launchpad Header.</p> <p>45. To do this, add the following code and modify the text in the setHeaderTitle function.</p> <p>46. In this code snippet it is important to note that the Users Full Name is an object already available in the Fiori Launchpad shell, hence we don't need to call any backend services and reuse the objects that are already existing in the Fiori Launchpad.</p>	<pre> //Get Date var today = new Date(); var dd = today.getDate(); var mm = today.getMonth(); var yyyy = today.getFullYear(); var text1 = dd.toString() + "/" + mm.toString() + "/" + yyyy.toString(); //Get Username var uname = sap.usHELL.Container.getUser().getFullName(); var text2 = "Welcome, " + uname; /** * Add item to the header */ rendererPromise.then(function (oRenderer) { oRenderer.setHeaderTitle(text2); }); </pre>

Steps / Comments	Screenshot
47. You can find references and documentation of these objects in the API Reference of SAPUI5	
<p>48. Your code should look like in the image.</p> <p>49. Once you have made the adjustments, Save your changes with the save button  and test the plugin in the Fiori Launchpad Sandbox</p>	 <pre> 14 init: function () { 15 var rendererPromise = this._getRenderer(); 16 17 // This is example code. Please replace with your implementation! 18 19 //Get Date 20 var today = new Date(); 21 var dd = today.getDate(); 22 var mm = today.getMonth(); 23 var yyyy = today.getFullYear(); 24 var text1 = dd.toString() + "/" + mm.toString() + "/" + yyyy.toString(); 25 26 //Get Username 27 var uname = sap.ushell.Container.getUser().getFullName(); 28 var text2 = "Welcome, " + uname; 29 /** 30 * Add item to the header 31 */ 32 rendererPromise.then(function (oRenderer) { 33 oRenderer.setHeaderTitle(text2); 34 }); 35 36 }, </pre>
50. You will now see a welcome message which includes the logged user's full name.	
<p>51. We have finished the warm-up, let's focus now on a real customer requirement.</p> <p>52. It is common for end-users to require information of the client number they are logged in.</p> <p>How would you solve this requirement?</p>	



55. ***Can I call a custom OData service to retrieve this information?***

You could, however, the best practice for Plugin development requires the objects to be independent of backend transactional data.

After all, you want this plugin to work on any Fiori Launchpad in any system so you should avoid dependencies to backend functionality

56. ***Can I call any of the Shell services to bring the current Client data?***

You can try searching in the [API Reference of SAPUI5](#) for a service or object that contains the current client information.

Note that by the time you finish searching and reading all the documentation you will most probably find nothing.



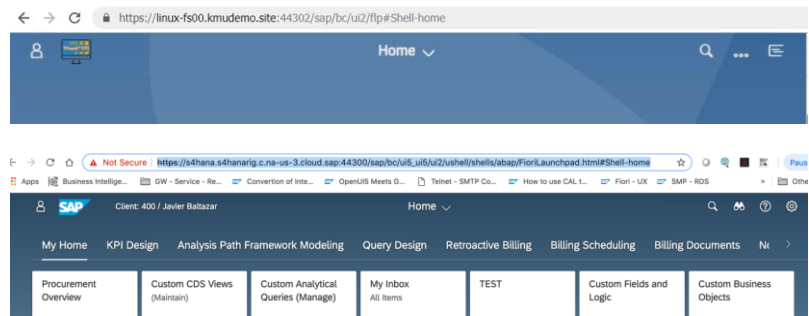
57. ***Can I use the User Default values in the Fiori Launchpad?***

You could, but unfortunately there is no parameter for a default client and again this means depending on backend functionality which is not desired for UI Plugins.

58. ***Can I read the URL parameters and display them in the Header text?***

You could, unfortunately, not all the calls to the Fiori Launchpad contain the client or language URL parameters.

Take a close look at the screenshots from two different systems. There is no guarantee that the URL parameters will be displayed.



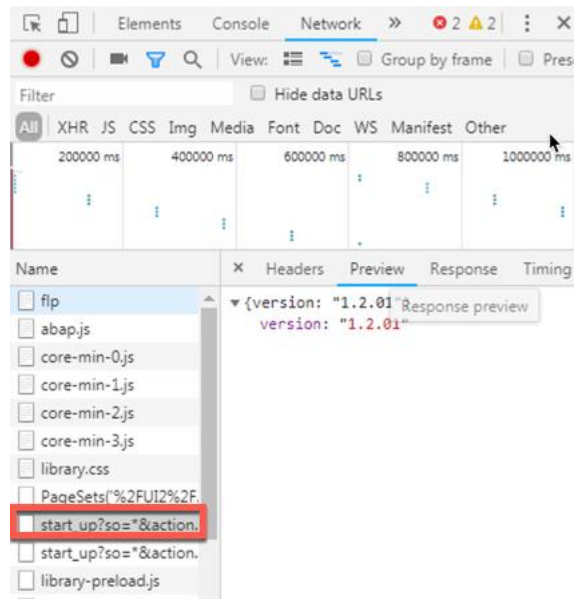


59. ***Can I simply hard code the values to my code? I doubt the end-users will like to have this information.***

Really...?

60. *I will use the UI2 “start-up” service call to retrieve the data*

Nice idea, but this will only work once your user ID has been assigned a Fiori Launchpad Catalog and Group. Meaning you will tie your Plugin to backend content availability, additionally, when there is no content assigned to your user, this service will return no data.



61. By now we hope you allow us to propose an idea as you may have now realized that despite being a simple number it is complex to obtain due to various restrictions, this is the main difficulty when building a UI Plugin.

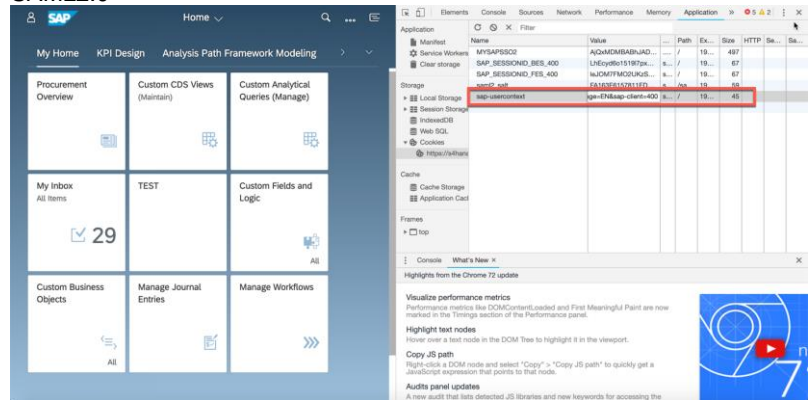
62. If you had time to analyze all the objects that are loaded when rendering the Fiori Launchpad you would notice that this value is not contained anywhere in the Fiori Launchpad structure.

63. You will find one possible solution to the problem if you load the Fiori Launchpad, navigate to the Application section and display the available cookies.

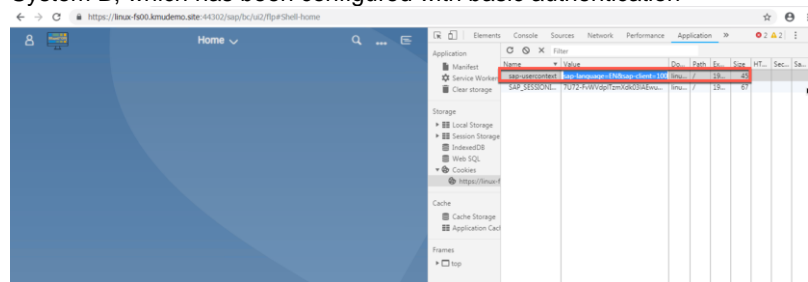
64. Both systems make use of the User Context cookie which contains information about the current logged in client.

Take a close look to the following different systems:

System A, which has been configured with advanced authentication using SAML2.0



System B, which has been configured with basic authentication



65. For the purpose of this exercise we will make use of this cookie as, in this moment, it is the only object available in the Fiori Launchpad that can help us solve the requirement.

66. Modify your plugin code.

```
//Get Date
var today = new Date();
var dd = today.getDate();
var mm = today.getMonth();
var yyyy = today.getFullYear();
var text1 = dd.toString() + "/" + mm.toString() + "/" +
yyyy.toString();

//Get Username
var uname = sap.ushell.Container.getUser().getFullName();
var text2 = "Welcome, " + uname;


//Get cookie context
var cname = "sap-usercontext";
var tmp = ";" + document.cookie;
var qk = tmp.split("; " + cname + "=");
var qk1 = qk[1].split("&");
var qk2 = qk1[0].split("&");
var qk3 = qk2[1].split("sap-client=");
var clnt = qk3[1].toString();
var text3 = "Client: " + clnt;

//Get client and username;
var text4 = text3 + " / " + uname;

rendererPromise.then(function (oRenderer) {
```

```
oRenderer.setHeaderTitle(text4);  
});
```

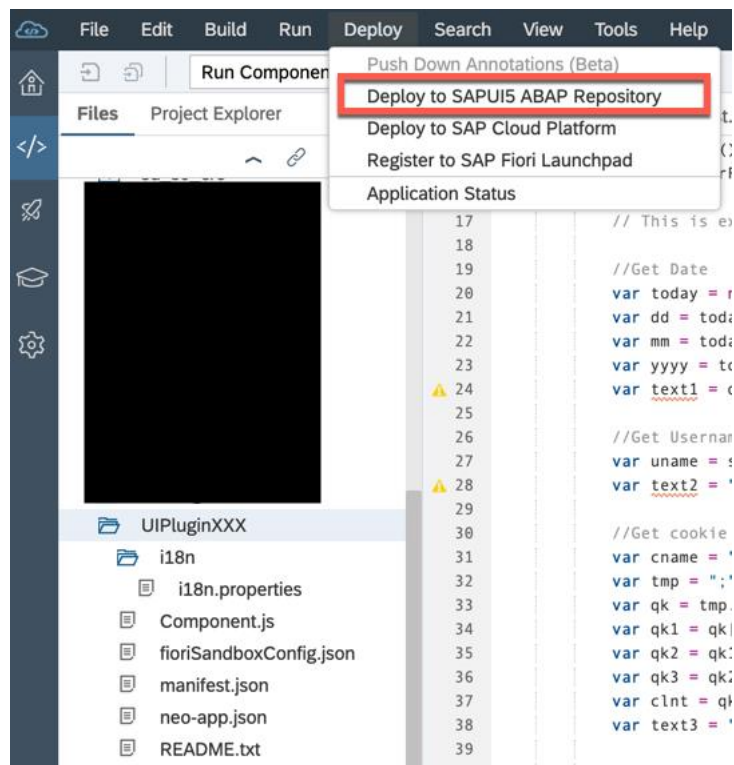
67. Your code should look like in the image.

68. Once you have made the adjustments, **Save** your changes with the save button .

69. You will not be able to test the Plugin with the Fiori Launchpad Sandbox as the cookies will not exist there, hence you need to deploy your Plugin to your ABAP system.

```
Component.js x manifest.json x  
14 + init: function () {  
15     var rendererPromise = this._getRenderer();  
16  
17     // This is example code. Please replace with your implementation!  
18  
19     //Get Date  
20     var today = new Date();  
21     var dd = today.getDate();  
22     var mm = today.getMonth();  
23     var yyyy = today.getFullYear();  
24     var text1 = dd.toString() + "/" + mm.toString() + "/" + yyyy.toString();  
25  
26     //Get Username  
27     var uname = sap.ushell.Container.getUser().getFullName();  
28     var text2 = "Welcome, " + uname;  
29  
30     //Get cookie context  
31     var cname = "sap-usercontext";  
32     var tmp = ";" + document.cookie;  
33     var qk = tmp.split("; " + cname + "=");  
34     var qk1 = qk[1].split("&");  
35     var qk2 = qk1[0].split("&");  
36     var qk3 = qk2[1].split("sap-client=");  
37     var clnt = qk3[1].toString();  
38     var text3 = "Client: " + clnt;  
39  
40     //Get client and username;  
41     var text4 = text3 + " / " + uname;  
42  
43 +     rendererPromise.then(function (oRenderer) {  
44         oRenderer.setHeaderTitle(text4);  
45     });  
46
```

70. To Deploy your code, select your project folder and in the tool bar navigate to **Deploy → Deploy to SAPUI5 ABAP Repository**



71. The deployment wizard will be displayed. Select your backend system ID and make sure the “Deploy a new application” checkbox is selected.

The screenshot shows the 'Deployment Options' dialog box with the following details:

- Deployment Options** (selected tab) | Deploy a New Application | Select a Transport Request | Confirm
- Deploy to SAPUI5 ABAP Repository**
- Deployment Options**
- *System**: FES - FES - RIG (dropdown menu)
- ☒ Deploy a new application
- ☐ Update an existing application
- Previous** | **Next**

72. Enter the details of your application and select a package where to deploy these objects.

73. For this exercise purpose we will use the following details:

Name: ZFIO_PLUGIN1

Description: Custom Plugin – FLP Header Title

****Note that the Name of your application will be the name of the deployed node in transaction SICF**

The screenshot shows the 'Deploy a New Application' dialog box with the following details:

- Deployment Options** | **Deploy a New Application** (selected tab) | Select a Transport Request | Confirm
- Deploy to SAPUI5 ABAP Repository**
- Deploy a New Application**
- Select a package for the application**
- *Name**: ZFIO_PLUGIN1
- *Description**: Custom Plugin - FLP Header Title
- *Package**: ZFIO_PLUGIN_DEMO | **Browse**
- Previous** | **Next**

74. Choose a Workbench Transport Request where to add the objects and click on Next.
75. Wait for the objects to be deployed in the ABAP system

Deployment Options

Deploy a New Application

Select a Transport Request

Confirm

X

Deploy to SAPUI5 ABAP Repository

Select a Transport Request

☒ Choose from requests in which I am involved

Request Number

BESK900471

Transport Request	User	Target	Description
BESK900465	I834429	/DEV/	Flori Workflow Demo
BESK900471	I834429	/DEV/	Flori Plugin Demo

☐ Create a new transport

Previous

Next

76. Once the deployment has finished, log in to the ABAP system and run transaction:

```
/UI2/FLP_CONF_DEF
```

77. In this transaction you will define the properties of your Plugin.
78. Create a new entry in the **Define FLP Plugins** section and enter the following details:

Plugin ID: ZFIO_PLUGIN1 (this should be a unique value in your system)

Description: Custom Plugin – FLP Header Title

UI5 Component ID: Component ID as described in the manifest.json file

URL: Path to node in transaction SICF

79. Save your entries and exit the transaction.

Table View Edit Goto Selection Utilities System Help

New Entries: Details of Added Entries

Dialog Structure

- Define FLP Property
- Define FLP Plugins
- Define Properties

FLP Plugin ID

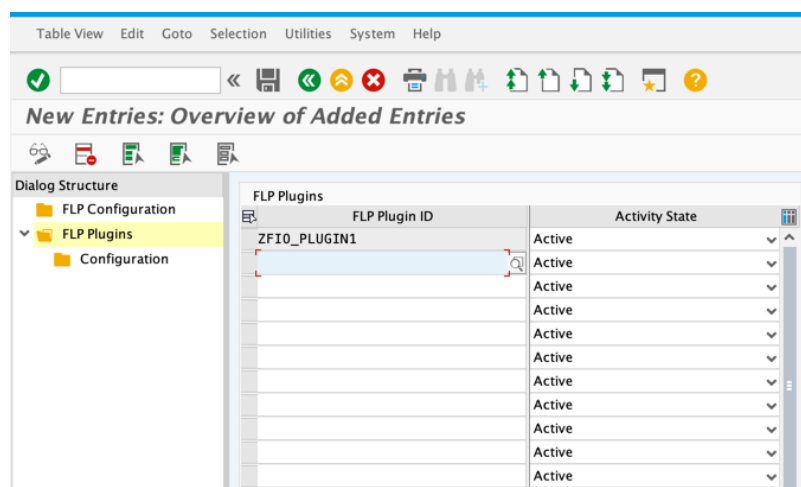
Define FLP Plugins

Description	Custom Plugin - FLP Header Title
UI5 Component ID	com.sap.rig.demo
URL	/sap/bc/ui5_ui5/sap/zfio_plugin1

80. To activate the Plugin, run transactions:

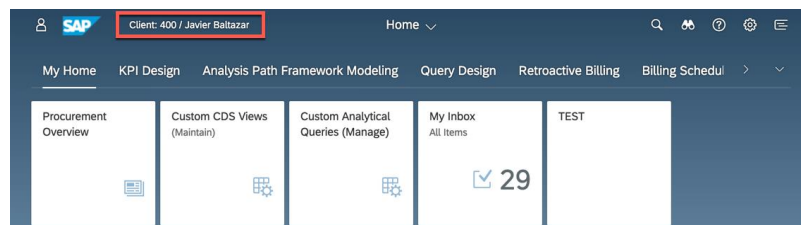
***/UI2/FLP_SYS_CONF → To make your plugin available system-wide
/UI2/FLP_CUS_CONF → To make your plugin available in specific clients***

81. Create a new entry in the **FLP Plugins** section and set your **Plugin ID as defined in the previous step.**
82. Make sure the status of your Plugin is set to active



FLP Plugin ID	Activity State
ZF10_PLUGIN1	Active
	Active
	Active
	Active
	Active
	Active
	Active
	Active
	Active
	Active
	Active
	Active

83. Once you have finished setting up the required configurations, use transaction **/UI2/FLP** to open your Fiori Launchpad and you should now see the header text being populated with the Client number and User Name.



Client: 400 / Javier Baltazar

Home

My Home KPI Design Analysis Path Framework Modeling Query Design Retroactive Billing Billing Schedule

Procurement Overview Custom CDS Views (Maintain) Custom Analytical Queries (Manage) My Inbox All items 29 TEST

www.sap.com/contactsap

© 2018 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.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

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. See <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.

