



User Guide | CONFIDENTIAL
2019-12-06

Extension Framework for SAP Trade Management, Version for SAP BW/4HANA 1.0

Content

- 1 Overview 4**
- 2 Common Activities 5**
 - 2.1 Extension Application Deployment. 5
 - Development of Extending UI5 Applications. 5
 - Including the Extension Files. 6
 - Deploying the Application as a BSP Application. 7
 - Deploying the Application as ABAP Upload. 12
 - 2.2 Retrieving UI5 Application Source Code Files from Repository. 14
 - 2.3 Maintaining Application Consistency. 18
- 3 Extension Tables 19**
 - 3.1 Namespace Table [extnamespaces]. 19
 - 3.2 View Extension Table [extviewextpoints]. 19
 - 3.3 View Extension Points. 22
 - Buyer Setup. 22
 - Fast Promotions. 22
 - Scenario Planning. 23
 - Common – Nav View. 23
 - Admin Area. 24
 - Detailed Promotion. 24
 - Promotion Landing Page. 27
 - Target Setting. 29
 - Target Setting Nav. 29
 - 3.4 Controller Extension Table [extcontroller]. 29
 - 3.5 Post-Load Extension Table [extpostloadfiles]. 30
- 4 Walkthrough – View Extension 32**
- 5 Walkthrough – Controller Extension 36**
- 6 Walkthrough – Post-Load Extension 39**
- 7 Walkthrough – Data Class Extension 43**
- 8 Walkthrough – i18n Extension 46**

Document History

Date	Comment
	Initial release for SAP Trade Management 1.0 FP02, version for SAP BW/4HANA

1 Overview

The aim of this user guide is to provide you with the information you need to extend the application by customizing extension tables, for example, `extnamespace` (a table that stores namespace-related customization), and deploying extended files in a UI5 extension application within the ABAP Repository.

Assumptions

This user guide is based on the assumption that you have some level of familiarity with the UI5 application and its components such as views, controllers, and fragments, and basic Web development.

→ Tip

We recommend that you refer to <https://sapui5.hana.ondemand.com/> if necessary when you extend UI5 controls, views, controllers, and fragments.

Prerequisites

Make sure that the following prerequisites are met before you customize and extend the application:

- **SAPGUI:** For maintaining customization in the SAP system.
- **IDE:** Any IDE or text editor, such as *Visual Studio Code*, *Eclipse*, *Notepad++*, *Sublime*, to edit UI5 source-related files; for example, XML, JS, JSON, and HTML.

i Note

SAPUI5 tools for Eclipse are no longer updated and are not available for download in higher Eclipse versions. However, if you still need to use it, consider the version based on plugins available. Recommended version based on customer feedback is [Eclipse 2018-12 Release](#) .

SAP Recommended Tool

SAP recommends using [Visual Studio Code](#) along with extended plugins available at [SAP Fiori Tools from Visual Studio Marketplace](#) .

The process of creating the extension files is similar to any other editor. The VS code has an advantage of JavaScript syntax and extensions for Fiori, which is useful in coding.

The code cannot be imported directly from the system, therefore, you need to download the BSP application code.

2 Common Activities

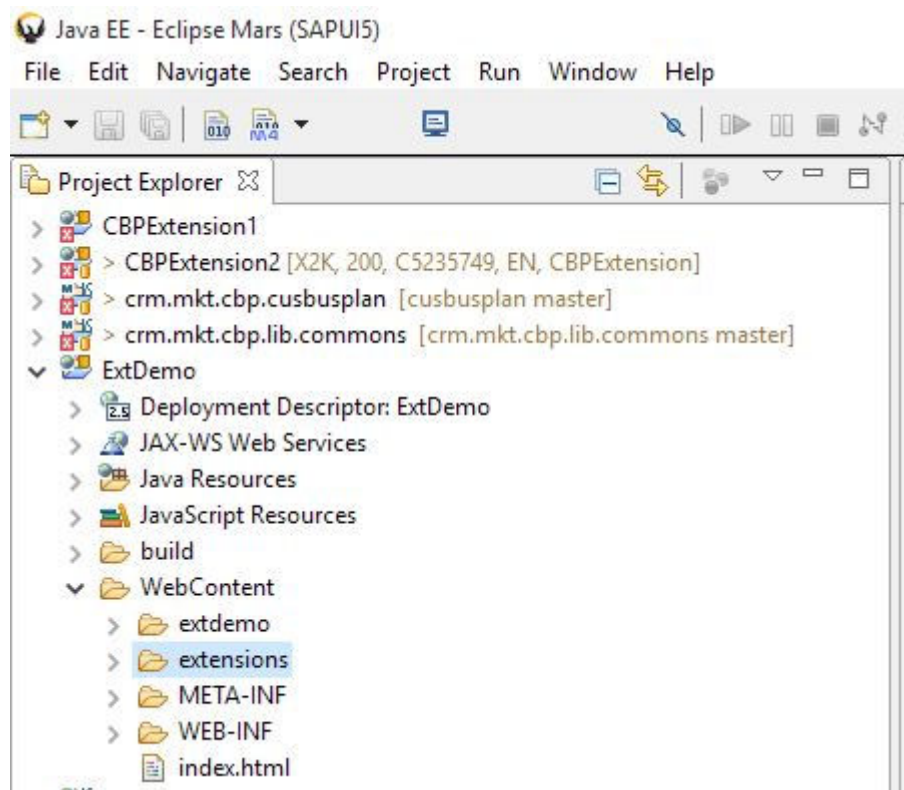
2.1 Extension Application Deployment

The activities that are required to implement each extension are listed in the following topics. The following examples are provided for reference only. You can refer to the table under [Extension Tables \[page 19\]](#) to locate the appropriate parameter values for the desired extension type; for example, a view extension.

2.1.1 Development of Extending UI5 Applications

The application source code contains the extension-related files, which include views, fragments, controllers, legacy controls, and data class-related files.

This application is deployed later as a BSP application in the ABAP Repository to provide the extension.



Java EE

2.1.2 Including the Extension Files

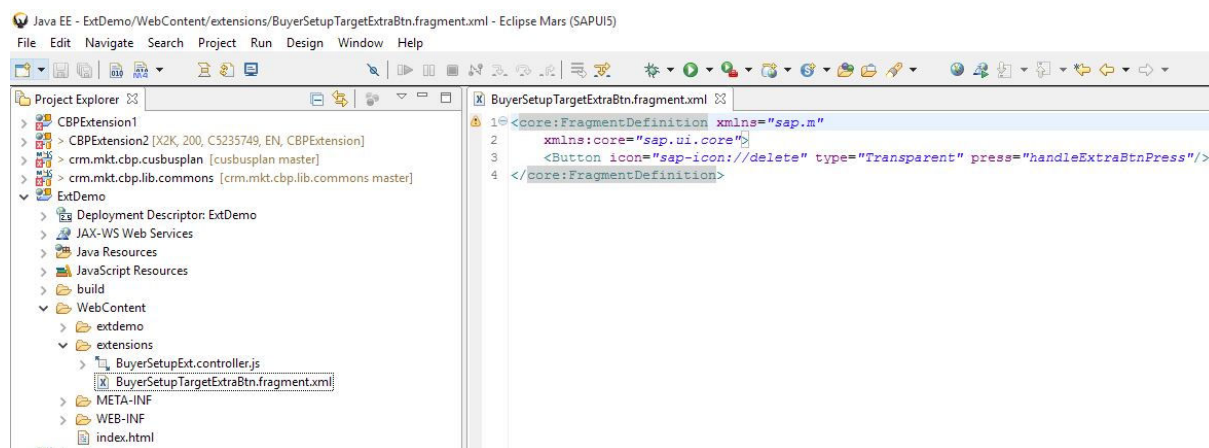
Extension-related files can be created inside a folder. The path to this folder should be provided during Customizing so that the framework looks for specified file in the settings provided.

Note

Throughout this document and the examples mentioned in this document, it is assumed that a folder named **extensions** is available that contains all of the relevant extended files.

Example

Let's assume you want to extend the *Buyer Setup* screen. You can create a fragment to render an additional button in the *Buyer Setup* screen as shown in the following figure.

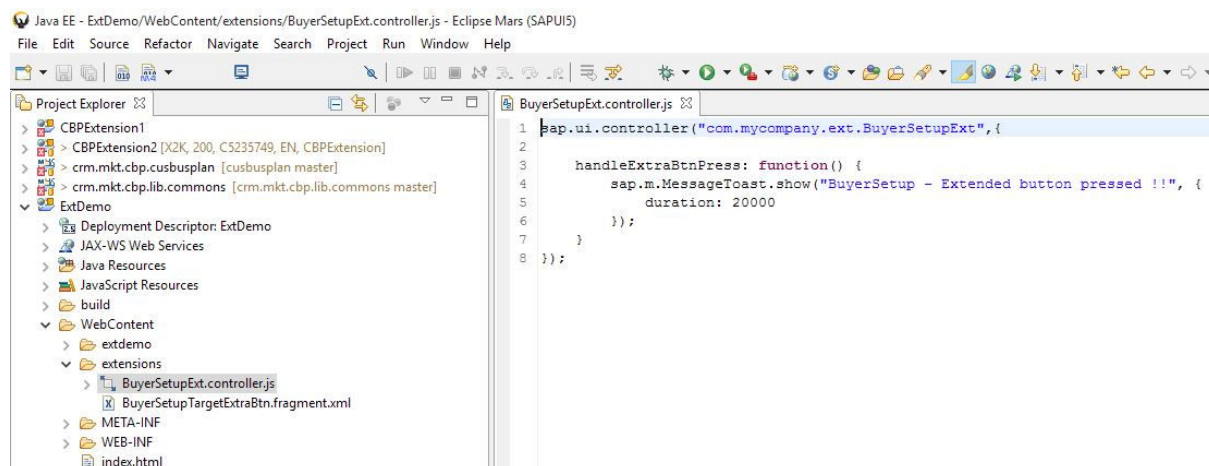


The screenshot shows the Eclipse IDE with the Project Explorer on the left and the XML editor on the right. The Project Explorer shows a project named 'ExtDemo' with a folder 'extensions' containing 'BuyerSetupExt.controller.js' and 'BuyerSetupTargetExtraBtn.fragment.xml'. The XML editor displays the following code:

```
1 <core:FragmentDefinition xmlns="sap.m"
2   xmlns:core="sap.ui.core">
3   <Button icon="sap-icon://delete" type="Transparent" press="handleExtraBtnPress"/>
4 </core:FragmentDefinition>
```

Java EE - ExtDemo1

You can also add an extended controller to include new handler methods or update existing methods as shown in the following figure.



The screenshot shows the Eclipse IDE with the Project Explorer on the left and the JavaScript editor on the right. The Project Explorer shows the same project structure as the previous figure, but with 'BuyerSetupExt.controller.js' selected. The JavaScript editor displays the following code:

```
1 sap.ui.controller("com.mycompany.ext.BuyerSetupExt", {
2
3   handleExtraBtnPress: function() {
4     sap.m.MessageToast.show("BuyerSetup - Extended button pressed !!", {
5       duration: 20000
6     });
7   }
8 });
```

Java EE - ExtDemo2

⚠ Caution

You should provide the appropriate namespace and exact path where the extension-related files are stored. You can also create subfolders for extension-related files regarding views, controllers, legacy controls, and data classes.

If you provide an incorrect name or path in the namespace table, the application may crash as it tries to search for extensions in a path that is either not present or not relevant.

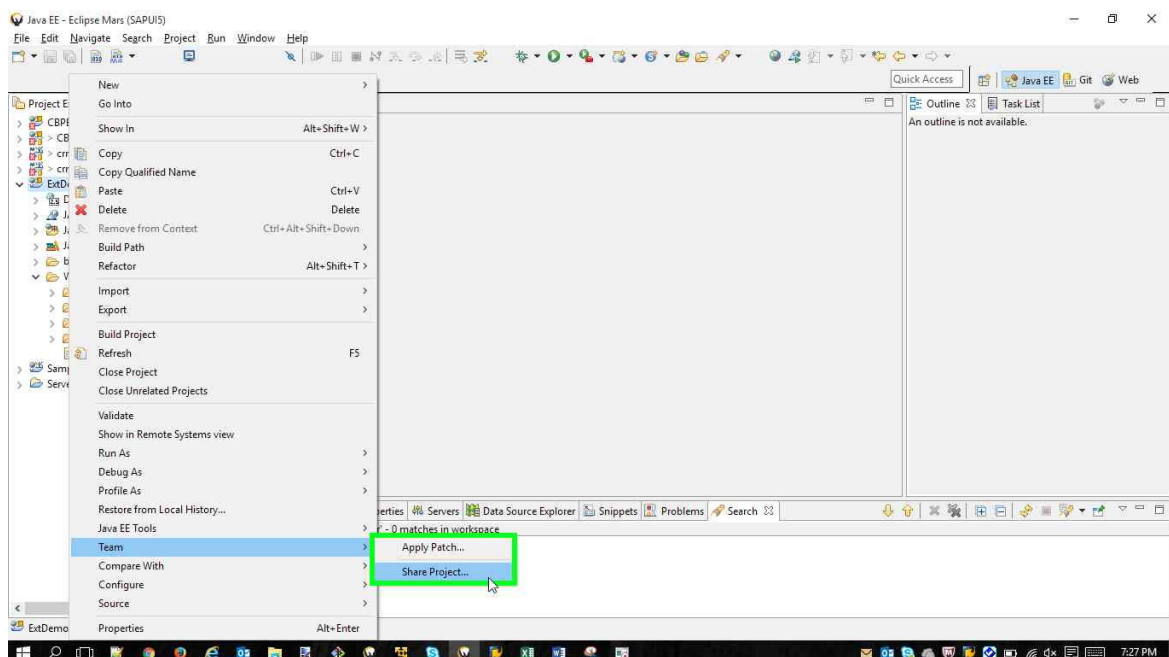
2.1.3 Deploying the Application as a BSP Application

Context

Once all of the extension-related files have been created for the extension, you can deploy the application in the ABAP Repository as shown in the following procedure.

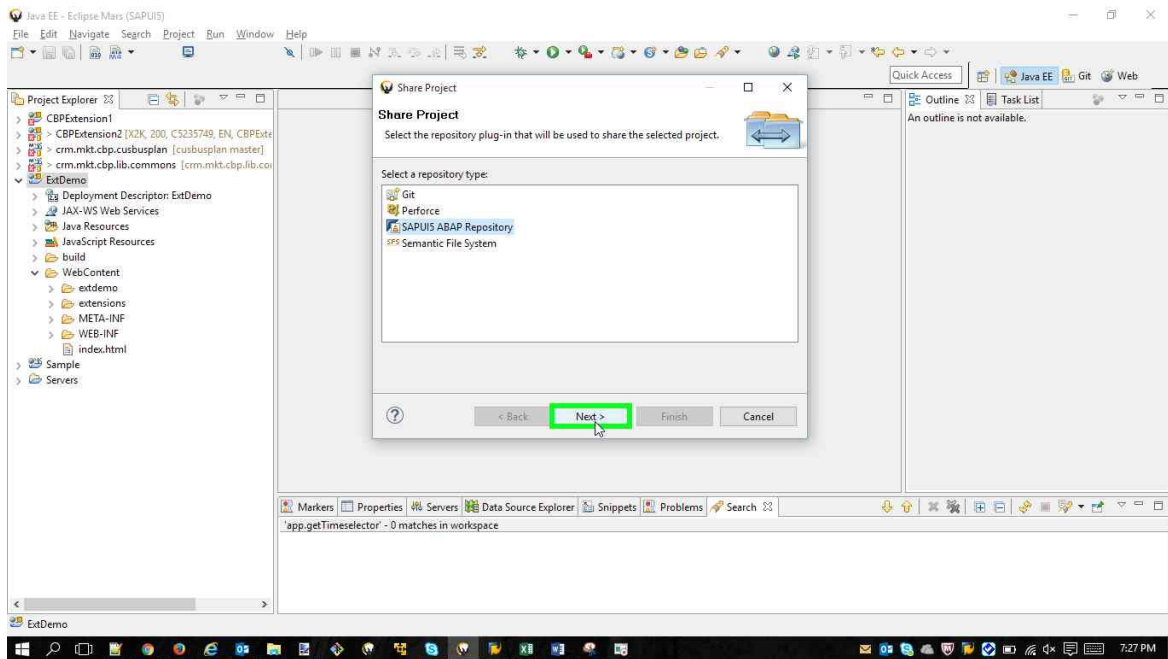
Procedure

1. Go to the *Project Explorer* view in *Eclipse*.
2. Right-click the extension project and choose **Team** > *Share Project*.



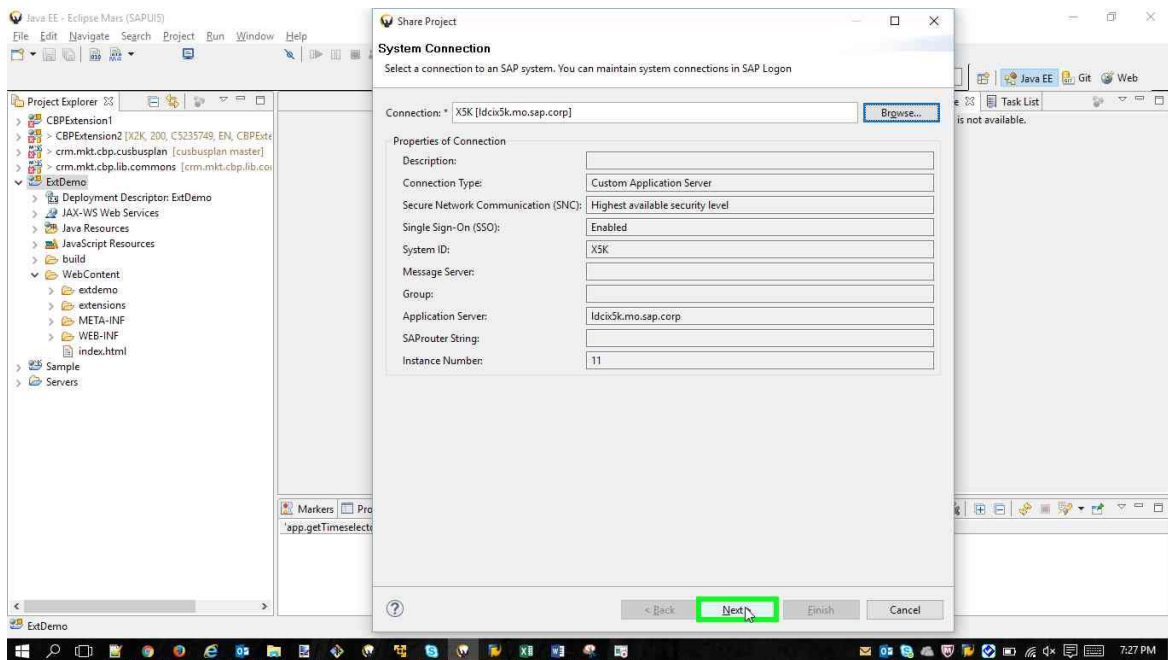
Share Project

3. Select *SAPUI5 ABAP Repository* and choose *Next*.



SAPUI5 ABAP Repository

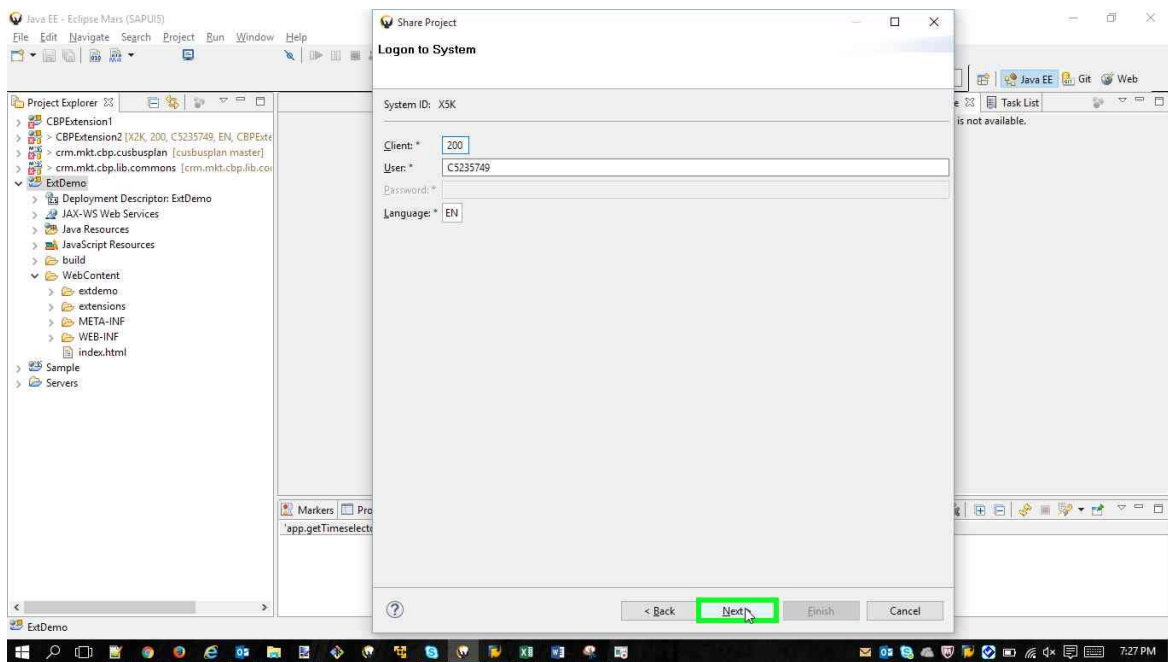
4. Browse and select the SAP system where the application is to be deployed and choose *Next*.



System Connection

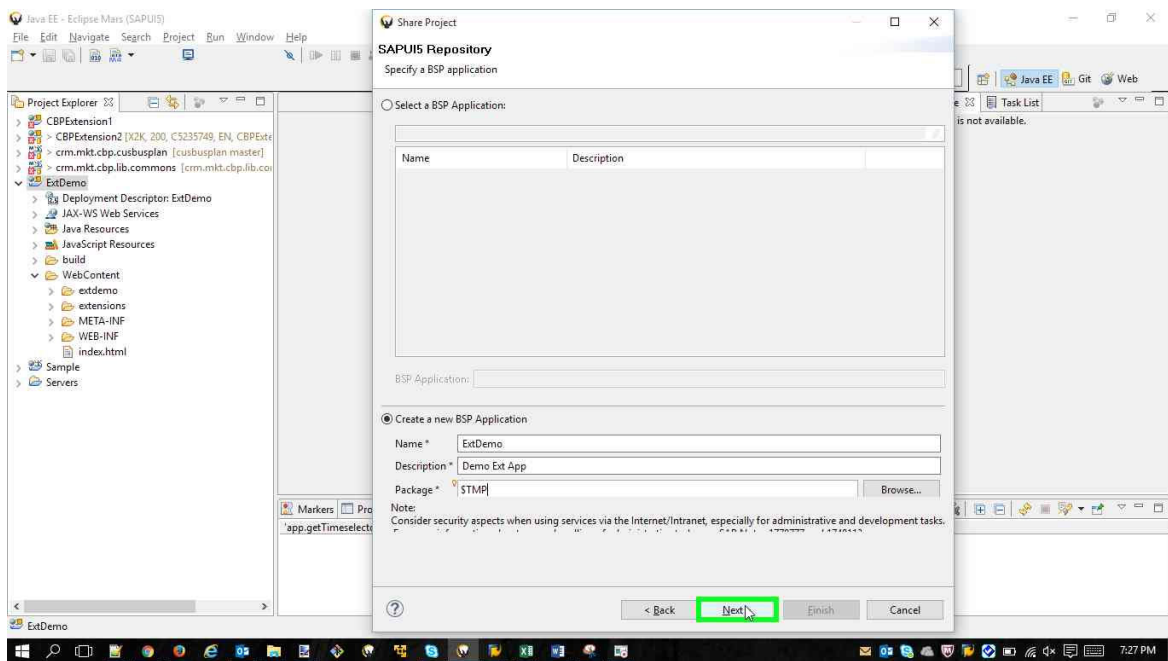
It is assumed that the user has the permissions required to deploy the application in the relevant SAP system.

5. Provide the appropriate client number and credentials and then choose *Next*.



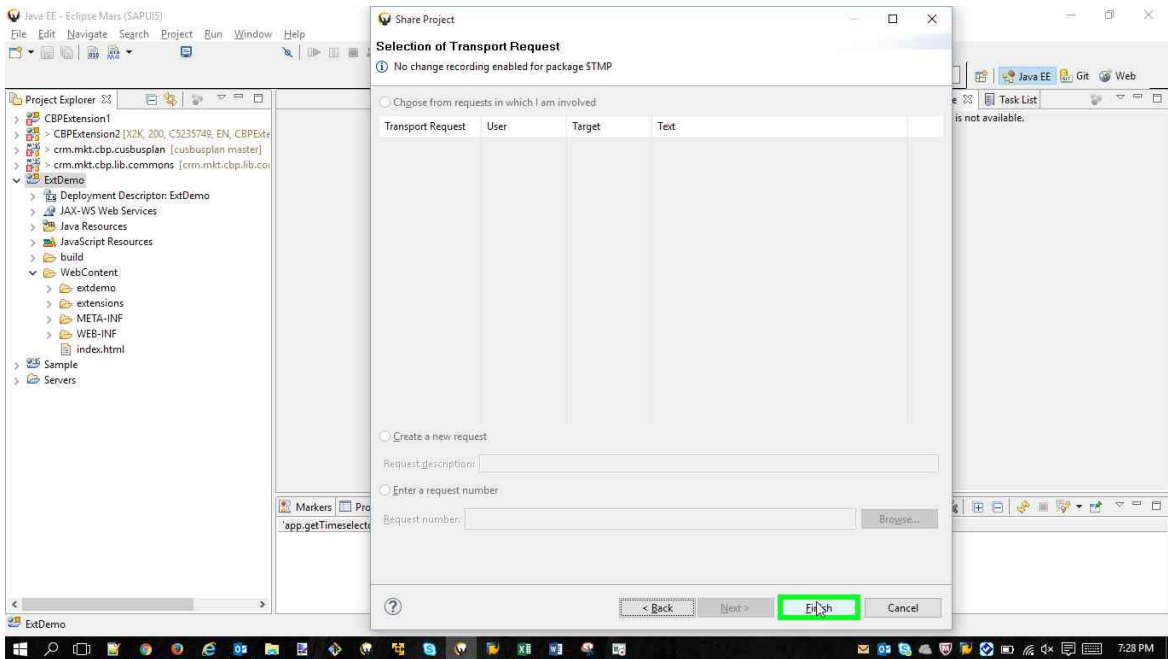
Client Number and Credentials

6. Enter the required details, such as the application name, description, and the package where the BSP application is to be deployed, then choose *Next*.



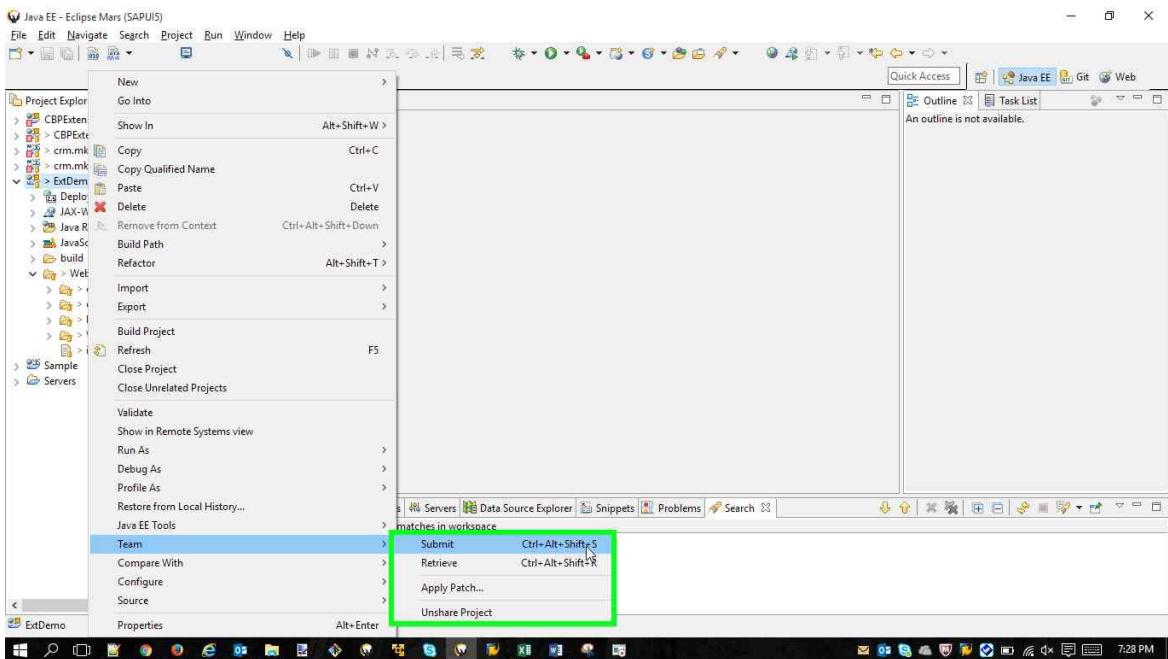
Create New BSP Application

7. Choose *Finish* to create an empty container for the application in the ABAP Repository with the specified details.



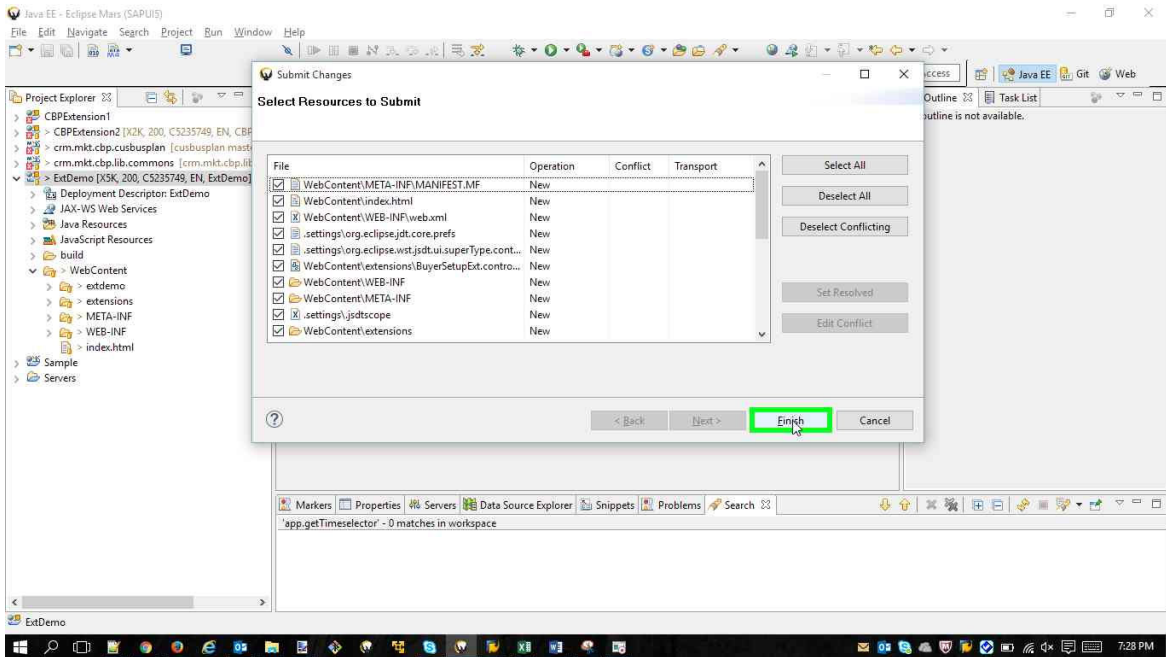
Select Transport Request

- Although the BSP application is now present in the specified package within the ABAP Repository, it does not have any extension-related files. You can now make the required changes and then submit them as shown in the following figure.



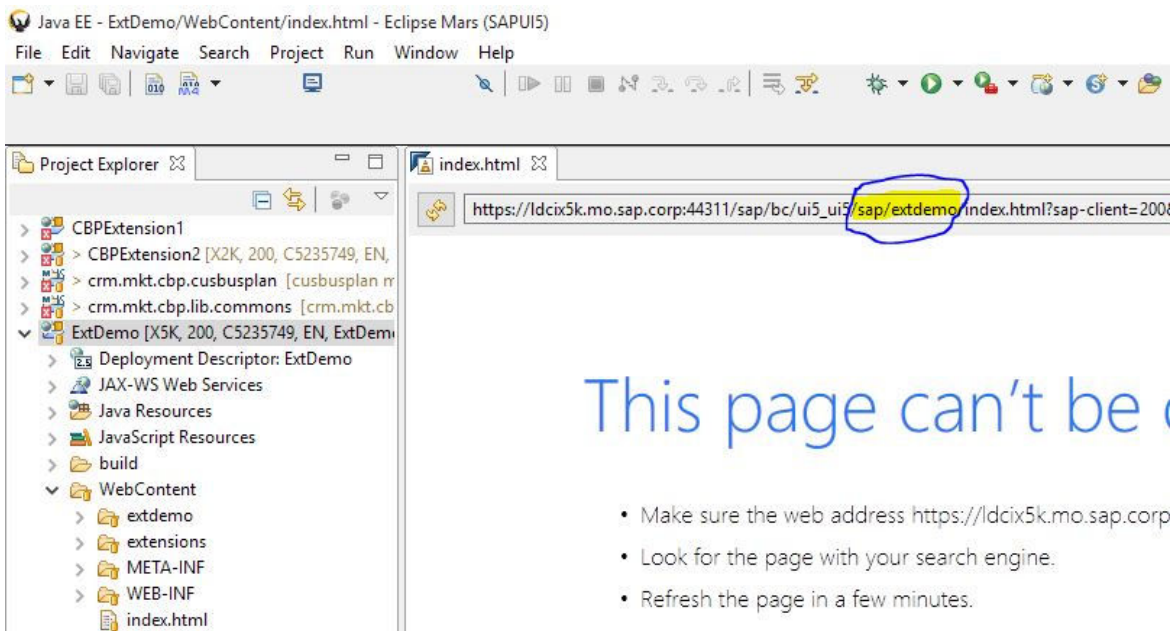
Submit

- Select all the files that are ready to be published and choose *Finish* to upload them.



Select Resources to Submit

- After the BSP application has been successfully uploaded, find the path where the application is deployed by running the application. A new browser tab opens and may render some content based on the `index.html` file or show an empty page. Note the relative path where the BSP application is deployed, as highlighted in the following figure.



This page can't be

- Make sure the web address `https://ldcix5k.mo.sap.corp`
- Look for the page with your search engine.
- Refresh the page in a few minutes.

Relative Path

⚠ Caution

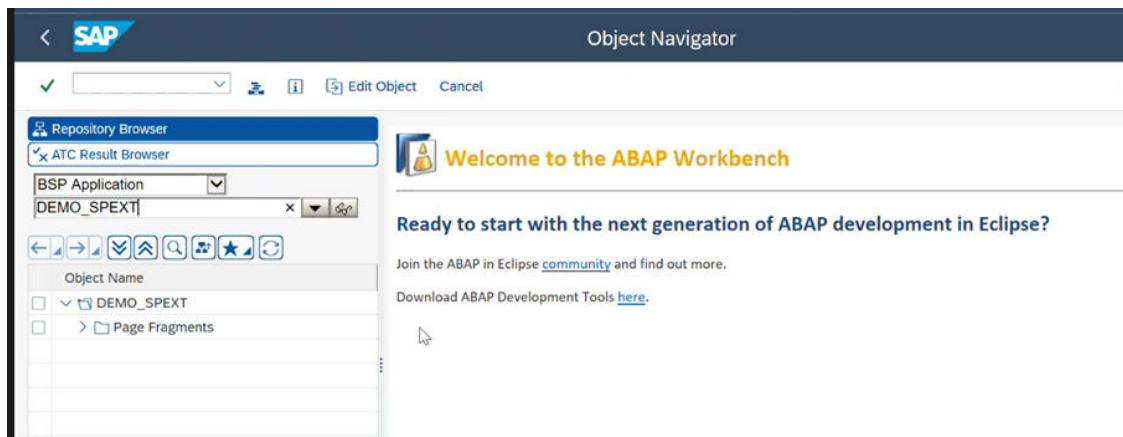
The relative path might be different to the example shown here. Make sure to carefully note the relative path only, as highlighted in the figure. This path must be provided in the Customizing settings for the

namespace. If you provide an incorrect name or path in the namespace table, the application may crash as it tries to search for extensions in a path that is either not present or not relevant.

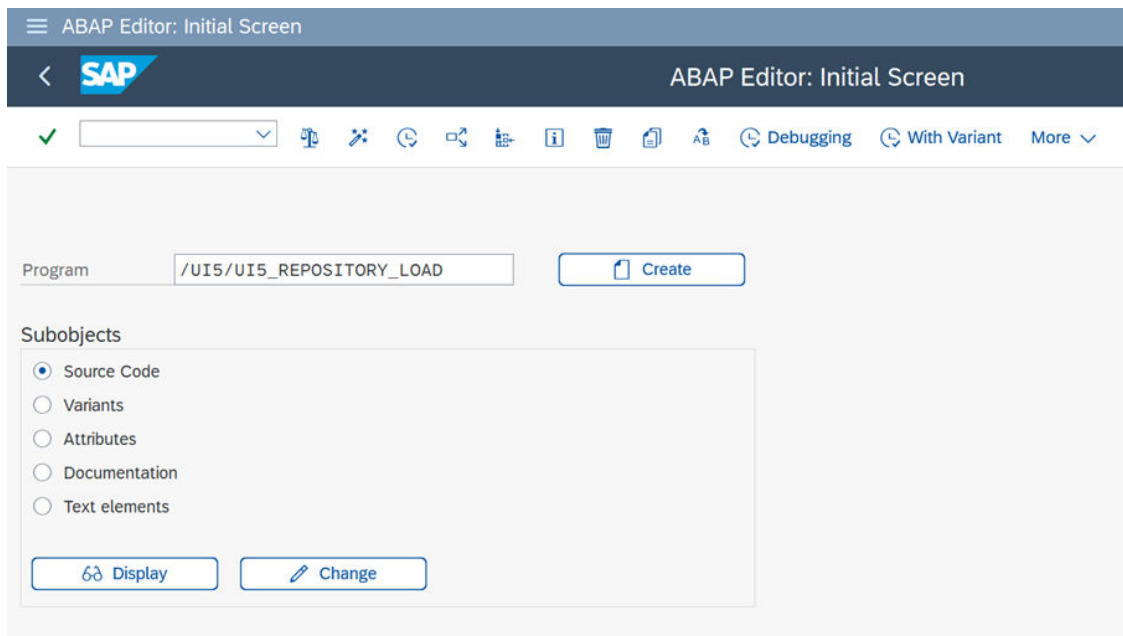
2.1.4 Deploying the Application as ABAP Upload

Deploying the application as ABAP upload involves the following tasks:

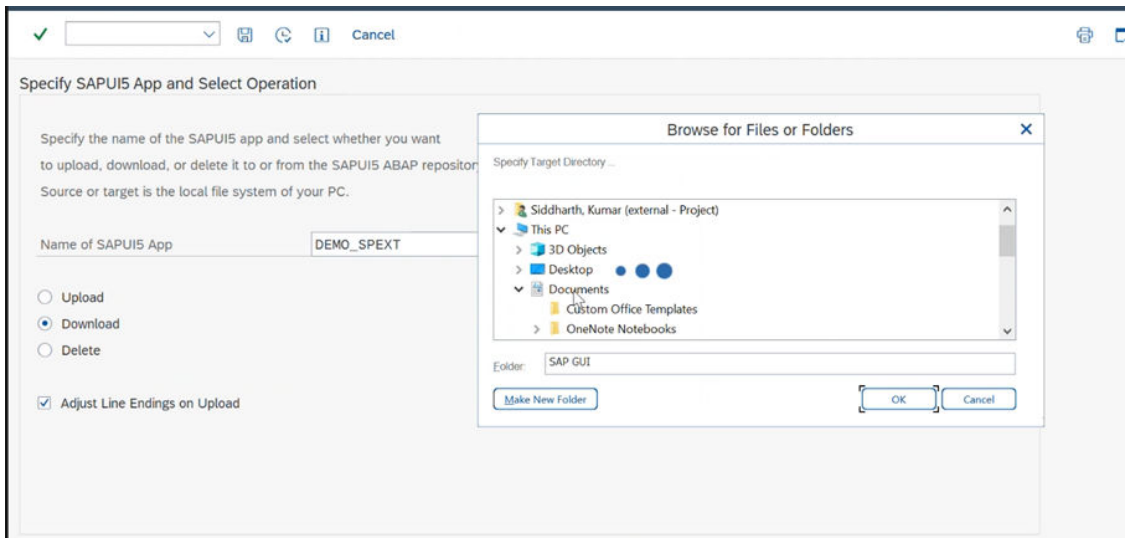
1. Download SAP UI5 application.
 - Access the Object Navigator.



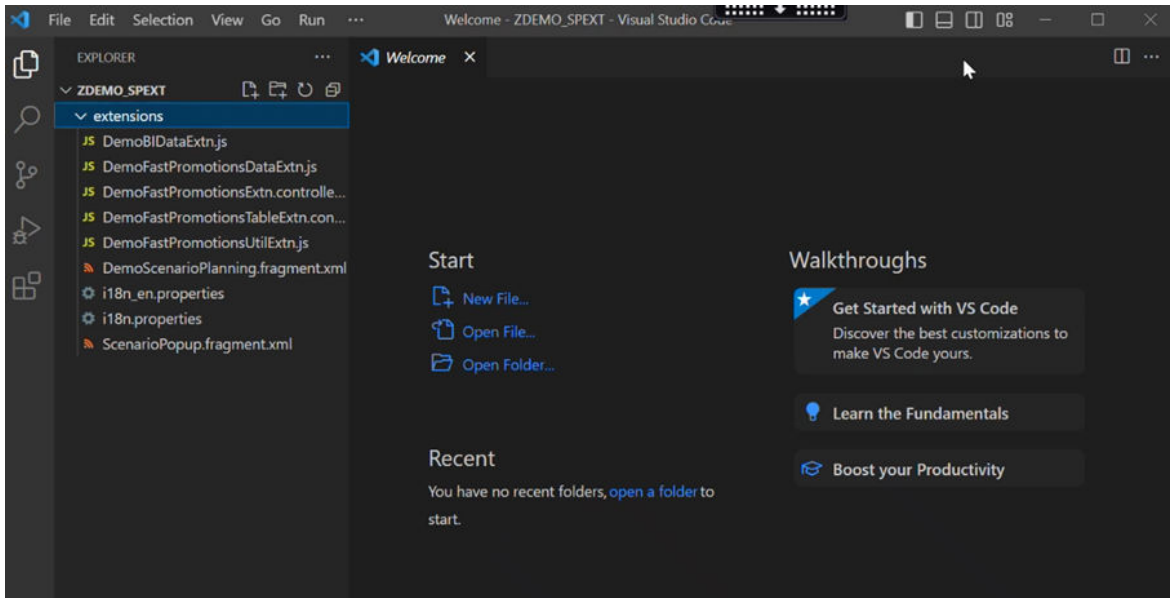
- Execute the SE38 transaction in ECC system and execute report `/UI5/UI5_REPOSITORY_LOAD`. Refer to [ABAP Upload/Download Report](#) for more information.



- Download the code for SAP UI5 BSP application.



2. Work with extensions via VS code.



Refer to the relevant topic in this guide for steps to create the extensions, and make the changes via the VS code.

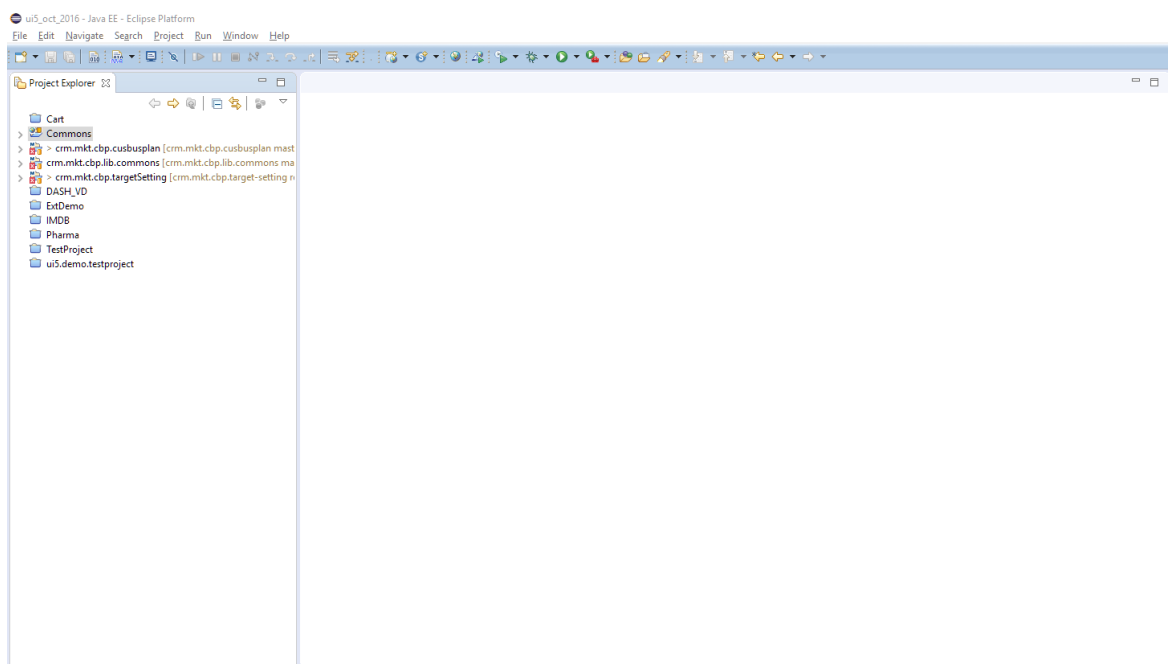
3. Use ABAP upload report for deployment

As Visual studio does not have an in-built feature for the deployment for SAP UI5 BSP application, use the deployment report to upload the code. Refer to [ABAP Upload/Download Report](#) for more information.

2.2 Retrieving UI5 Application Source Code Files from Repository

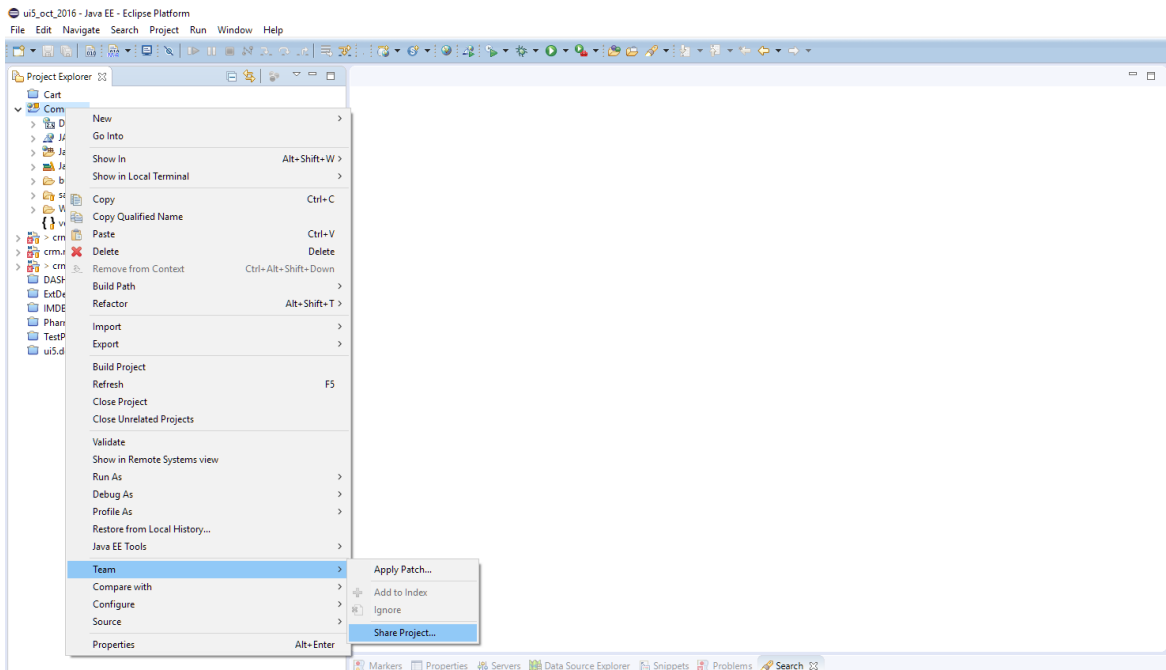
Procedure

1. Create a UI5 application that would contain the application source code files after syncing with the ABAP repository.



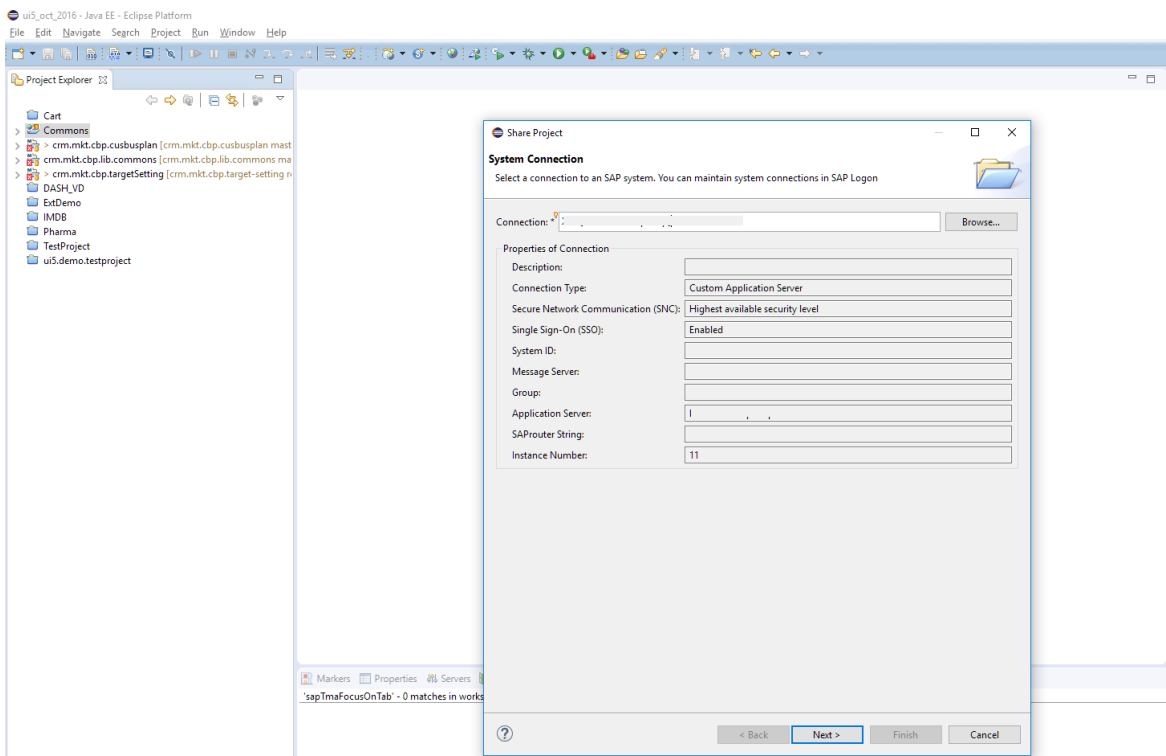
Create UI5 Application

2. Right-click the new project and choose **Team > Share Project...** from the context menu.



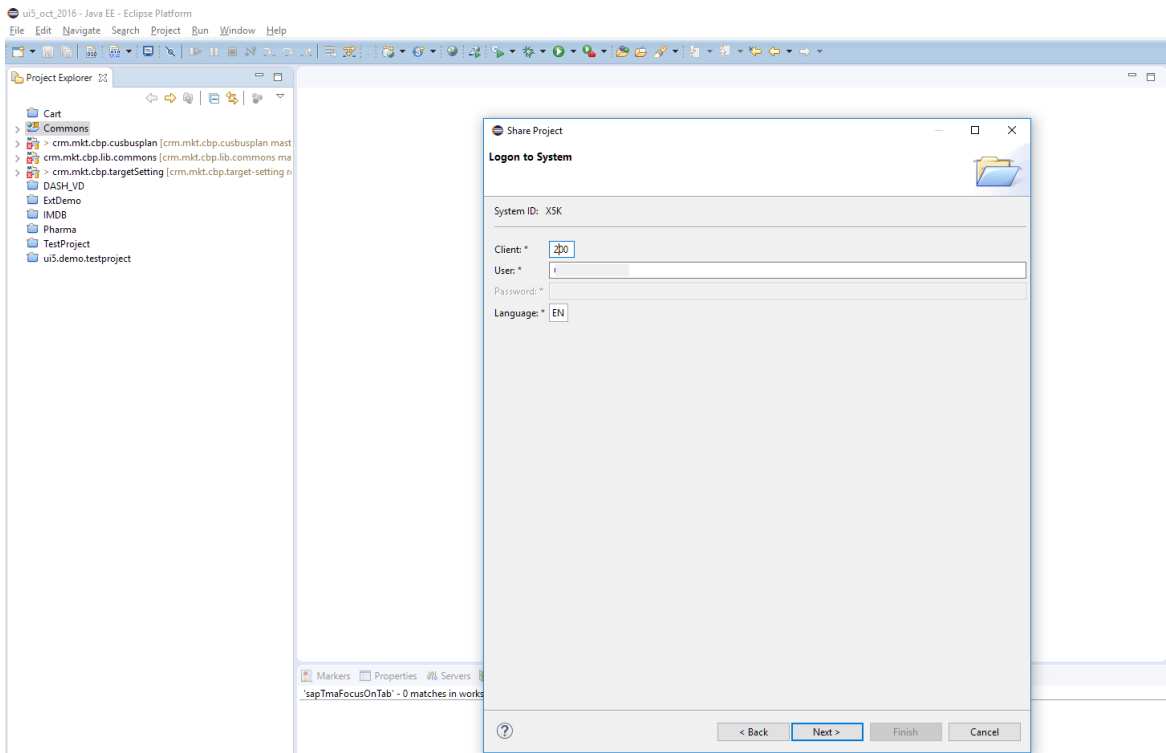
Share Project

3. Enter the system ID, select the appropriate system from the list of suggestions, and then choose *Next*.



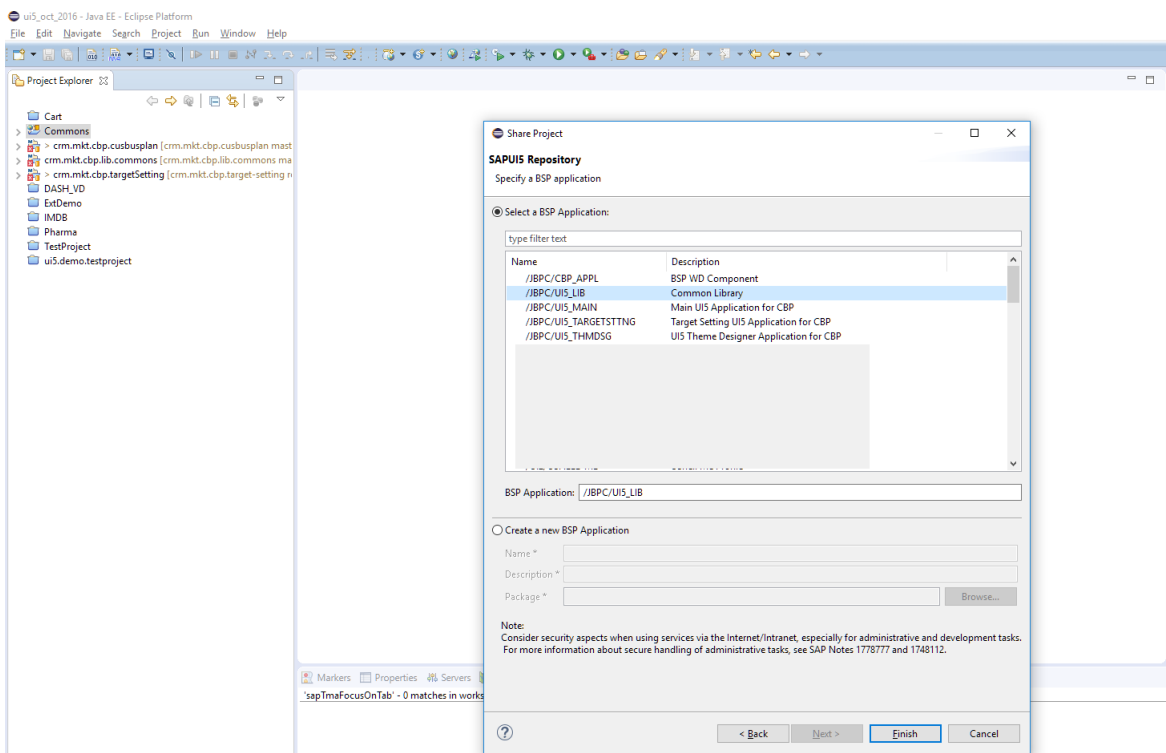
System Connection

4. Enter the appropriate client number and your credentials to log on, then choose *Next*.



Logon to System

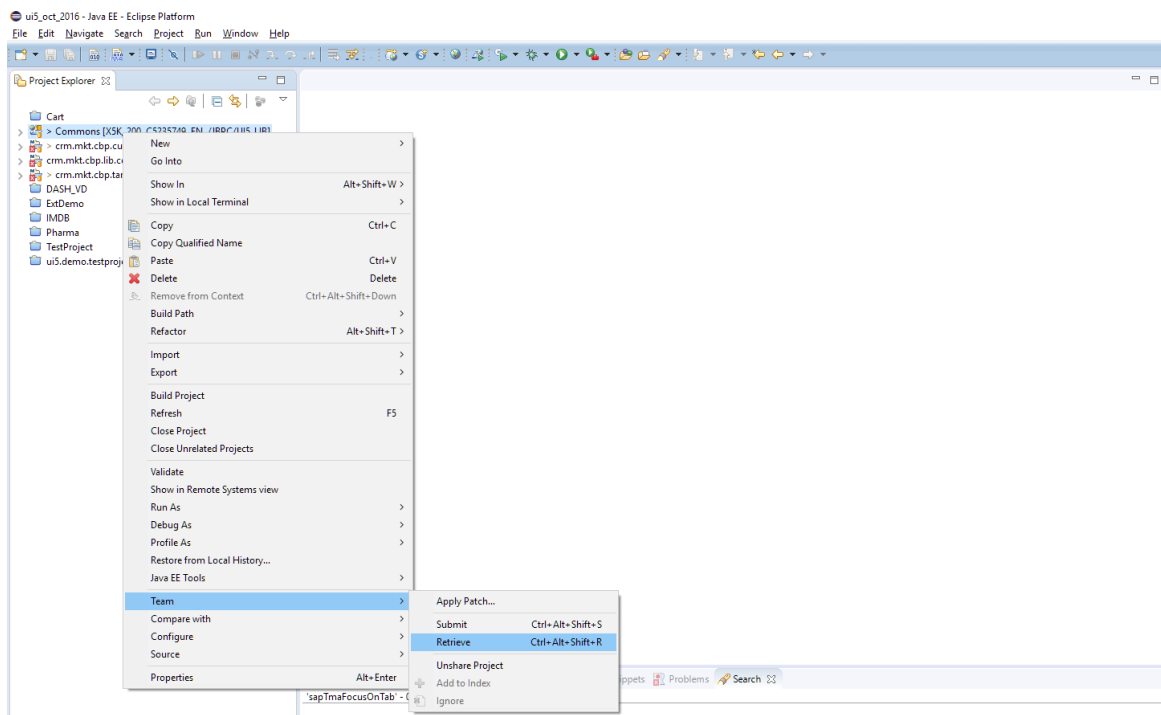
5. Select the BSP application name, which in our case is /JBPC/UI5_LIB.



BSP Application Name

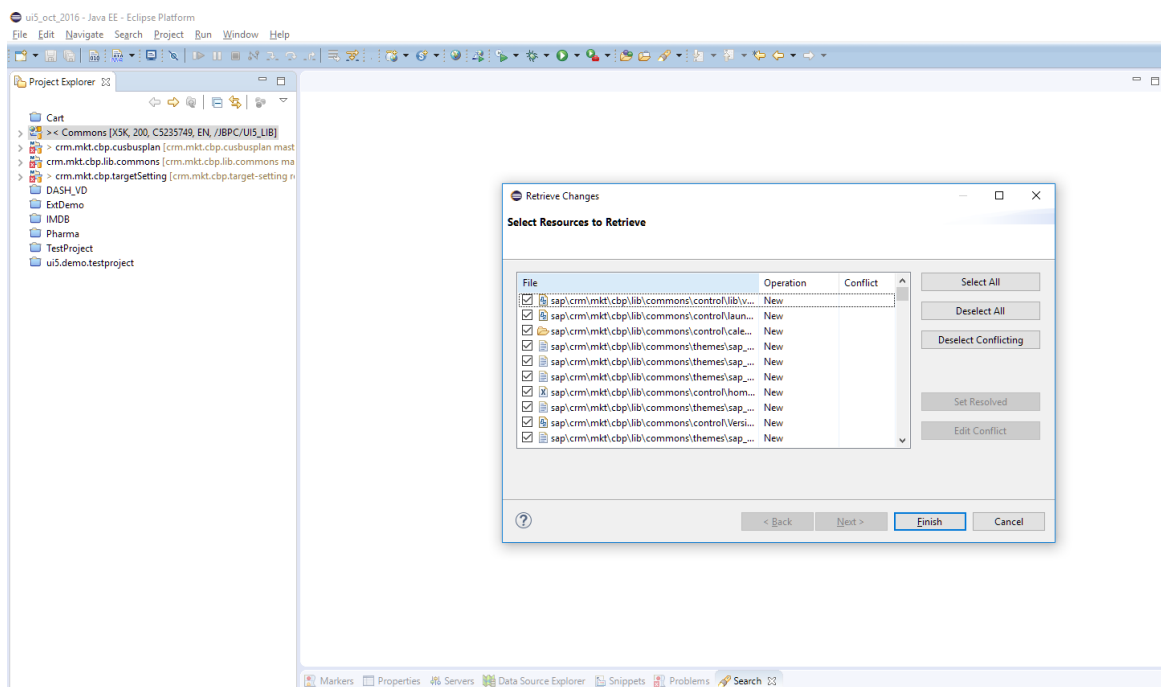
The project is now linked to the BSP application and can be retrieved.

- Retrieve the application by choosing **Team > Retrieve**.



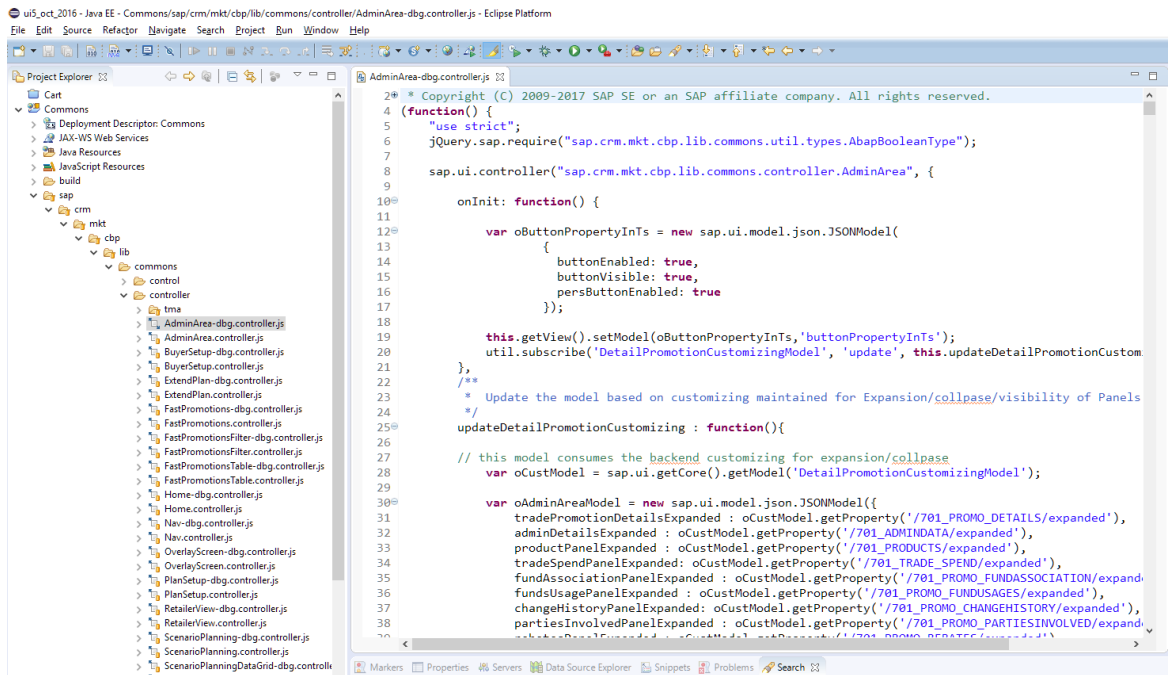
Retrieve the Application

- Select the files you wish to download or update, then choose *Finish*.



Select Files to Download

Source code files get downloaded inside the project. The files ending with **-dbg** are unminified files.



Source Code Files

2.3 Maintaining Application Consistency

You must maintain application consistency if you have used the extension framework and implemented a UI-related SAP Note.

Prerequisites

You are using the extension framework and have implemented an SAP note for the user interface.

Context

If you used the extension framework for any kind of extension, after each UI SAP note implementation you must test the extended area of the application.

A UI-related SAP Note may bring in new features that may conflict with the changes made by using the UI extension framework. The application has many reusable code blocks; therefore, it's imperative to revisit and ensure that the expected functionality is not affected by the new change.

3 Extension Tables

The tables that can be referred to for maintaining the Customizing settings are listed in the following topics.

You can maintain settings for extension tables in Customizing for *Customer Relationship Management* under [▶ Trade Management ▶ General Settings ▶ UI Extension ▶](#).

3.1 Namespace Table [extnamespaces]

The Customizing activity for settings related to the namespace table is *Maintain Extension Namespace*. The extension project has to be deployed in the ABAP Repository and the deployment path should be provided in this Customizing activity. Provide the module name and the relative path for the extension project deployed in the ABAP Repository. The extension framework searches for extended files in this relative path.

→ Remember

The namespace Customizing settings are crucial for the extension framework, as it searches for the extended files at the location provided for the namespace. The extension framework maps the provided namespace with the given path and looks for the file in this path whenever a `require` statement with matching namespace is executed.

Maintain the following settings in Customizing:

Extension Namespace	Extension Path
com.mycompany.ext	../../../../sap/extdemo/extensions

i Note

The example above illustrates a namespace with a relative path. The actual location might differ and the path could be subdirectory of any other directory. You can retrieve the relative path while testing the deployed BSP application.

3.2 View Extension Table [extviewextpoints]

The Customizing activity for settings related to the view extension table is *Define View Extension Points*.

You can find the relevant extension information for your requirements in the following table.

Choose the *View Name* and *Extension Point Name* for the *Required Extension*.

For more information, see [Walkthrough – View Extension \[page 32\]](#).

Required Extension	View Name	Extension Point Name
Buyer Setup – Add Buyer Form Field	sap.crm.mkt.cbp.lib.commons.view.BuyerSetup	extBuyerSetupFormField [page 22]
Buyer Setup – Target Btn		extBuyerSetupBuyerTargetBtn [page 22]
Fast Promotions – Toolbar Item	sap.crm.mkt.cbp.lib.commons.view.FastPromotions	extFastPromotionsToolbarItem [page 22]
Scenario Planning – Toolbar Item	sap.crm.mkt.cbp.lib.commons.view.ScenarioPlanning	extScenarioPlanningToolbarItem [page 23]
Common – Header Nav Bar Right Item	sap.crm.mkt.cbp.lib.commons.view.Nav	extHeaderNavbarRightItem [page 23]
Common – Header Top Bar Right Item		extHeaderTopbarRightItem [page 23]
Common – Header Top Bar Left Item		extHeaderTopbarLeftItem [page 23]
Detailed Promotion Admin Area – Trade Promotions Details General Data	sap.crm.mkt.cbp.lib.commons.view.AdminArea	extAdmAreaTPDetailsGeneralData [page 24]
Detailed Promotion Admin Area – Trade Promotions Additional		extAdmAreaTPDetailsAdditionalData [page 24]
Detailed Promotion Admin Area – Trade Promotions Sales Org		extAdmAreaTPDetailsSalesOrgData [page 24]
Detailed Promotion Admin Area – Administration General		extAdmAreaAdministrationGeneralData [page 24]
Detailed Promotion Admin Area – Administration Integration		extAdmAreaAdministrationIntegrationData [page 24]
Detailed Promotion – General Data Fields	sap.crm.mkt.cbp.lib.commons.view.tma.DetailedPromotion	extDetailedPromoGenData [page 24]
Detailed Promotion – Forms in Trade Promotion Details		extDetailedPromoDateFields [page 24]
Detailed Promotion – Footer Bar Buttons		extDetailedPromoFooterBarButton [page 25]
Detailed Promotion – Volume Planning Segment Button	sap.crm.mkt.cbp.lib.commons.view.tma.VolumePlanning	extDetailedPromoVVPSegmentButton [page 25]

Required Extension	View Name	Extension Point Name
Detailed Promotion – Volume Planning Panel Button		extDetailedPromoVPPanelButton [page 25]
Detailed Promotion – Promotion Lifecycle Area	<code>sap.crm.mkt.cbp.lib.commons.fragment.tma.DetailedPromotionHeader</code>	extDetailedPromoPromoLifeCycleArea [page 25]
Landing Page – Between Fast Promotion Table and Header	<code>sap.crm.mkt.cbp.lib.commons.fragment.tma.PromotionsLandingPageHeader</code>	extLandingPageFastPromoTableHeader [page 27]
Fast Promotion Filter	<code>sap.crm.mkt.cbp.lib.commons.view.FastPromotionsFilter</code>	extFastPromotionFilter [page 27]
Detailed Promotion – Tactics Popup	<code>sap.crm.mkt.cbp.lib.commons.view.tma.Tactics</code>	extTacticsPopUp [page 26]
Detailed Promotion – Trade Spends Dialog	<code>sap.crm.mkt.cbp.lib.commons.fragment.tma.DetailPromotionTradeSpends</code>	extTradeSpendsDialog [page 26]
Landing Page – Mass Copy with Time Shift Dialog	<code>sap.crm.mkt.cbp.lib.commons.fragment.BulkActionsMassCopyDialog</code>	extMassCopyDialog [page 28]
Landing Page – Mass Change Dialog	<code>sap.crm.mkt.cbp.lib.commons.fragment.BulkActionsMassChangeDialog</code>	extMassChangeDialog [page 28]
ITP – Table Toolbar Left Items	<code>sap.crm.mkt.cbp.targetSetting.view.TargetSetting</code>	extITPTableToolbarLeft [page 29]
ITP – Table Toolbar Right Items		extITPTableToolbarRight [page 29]
ITP – Footer Bar Right Items	<code>sap.crm.mkt.cbp.targetSetting.view.Nav</code>	extFooterBarItem [page 29]
ITP – Nav Toolbar Right Items		extNavToolbarRightItem [page 29]
ITP – Sub Toolbar Right Items		extSubToolbarRightItem [page 29]
ITP – Nav Header Right Items		extHeaderRightItem [page 29]

3.3 View Extension Points

3.3.1 Buyer Setup

Plan: J-00001005

SAP Mart Potsdam Hyper 2016 Operational Plan

Plan Overview **Plan Setup** Assortment Plan Scenario Planning

Buyer Setup Planning Product Hierarchy

Internal Targets

Sales Volume

181.818,1818 CS

Buyers +

jack hughman

Roger Linda

Test Buyer

Name: jack hughman

E-Mail:

Phone Number:

Cell Phone Number:

extBuyerSetupFormField

Set Buyer's Targets 2016 extBuyerSetupBuyerTargetEbn Copy From Previous Year +

View Extension Points for Buyer Setup

3.3.2 Fast Promotions

5 Promotions 5 Approval Standard								
Actions	ID/Description	Type	Plan Start/End Date	Customer	Tactic	Status	Buying Start/End Date	
<input type="checkbox"/>	T-00018714 No Description	Detailed Pro...	04.12.2017 - 10.12.2017	SAPMart Narvik / 4...	Media Support 9 More	New	04.12.2017 - 10.12.2017	
<input type="checkbox"/>	T-00018712 No Description	Detailed Pro...	01.11.2017 - 07.11.2017	SAPMart Narvik / 4...	POS 9 More	Rejected	01.11.2017 - 07.11.2017	
<input type="checkbox"/>	T-00018711 No Description	Detailed Pro...	25.09.2017 - 12.11.2017	SAPMart Narvik / 4...	POS 9 More	Planned	18.09.2017 - 05.11.2017	
<input type="checkbox"/>	T-00016448 Sept Beverages Promo	Draft Promot...	01.09.2017 - 30.09.2017	SAPMart Narvik / 4...		Draft	01.11.2017 - 30.11.2017	

View Extension Points for Fast Promotions

3.3.3 Scenario Planning

Plan: J-00001005

SAP Mart Potsdam Hyper 2016 Operational Plan

Plan Overview Plan Setup Assortment Plan Scenario Planning

Profit & Loss Product Tree

2016 New

Quarter1 CW 53-13 Quarter2 CW 13-26 Quarter3 CW 26-39 Quarter4 CW 39-52

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

Planned Scenarios

Product Units Update Plan/Scenario

Scenario Comparison

KPIs Operational Plan

No data

223 Promotions 223 Planned Standard

Search

Recalculate Save Cancel

View Extension Points for Scenario Planning

3.3.4 Common – Nav View

Plan: J-00001005

SAP Mart Potsdam Hyper 2016 Operational Plan

ext-HeaderTopbarLeftItem ext-HeaderTopbarRightItem ext-HeaderNavbarRightItem

Plan Overview Plan Setup Assortment Plan Scenario Planning

Profit & Loss Product Tree

2016 New

Quarter1 CW 53-13 Quarter2 CW 13-26 Quarter3 CW 26-39 Quarter4 CW 39-52

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

Planned Scenarios

Product Units Update Plan/Scenario

Scenario Comparison

KPIs Operational Plan

No data

223 Promotions 223 Planned Standard

Search

Recalculate Save Cancel

View Extension Points for Common – Nav View

3.3.5 Admin Area

The screenshot displays the SAP Admin Area configuration page for 'Trade Promotion Details'. The main content area is divided into two sections: 'General Data' and 'Administration'. The 'General Data' section includes fields for Customer Plan ID (J-00000956), Employee Responsible (Mr. and Mrs. Rajat Chauwdhary), Planning Profile Group (CBP Planning on Prd&BPHN D...), Funds Plan, Agreement, Validation Profile, Sales Organization (CBP MIT Test), and Distribution Channel (Final customer sales). The 'Administration' section includes fields for Changed By (C5248037), Created By (C5240106), and Integration options (Created in BW, Created in ERP). Extension points are highlighted in yellow: 'extAdmAreaTPDetailsGeneralData', 'extAdmAreaTPDetailsAdditionalData', and 'extAdmAreaAdministrationIntegrationData'. The right sidebar shows a summary for 'Plan: SAP Mart Norway North 2016' with Sales Volume (-134.89) and ROI (-102.02%). A 'Recalculate' button is visible at the bottom right.

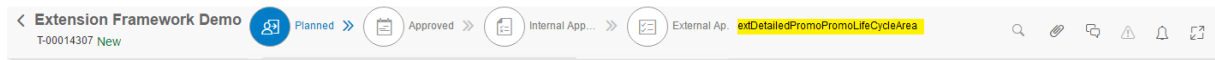
View Extension Points for Admin Area

3.3.6 Detailed Promotion

The screenshot displays the SAP Detailed Promotion configuration page. The main content area is divided into two sections: 'General Data' and 'Tactics'. The 'General Data' section includes fields for Customer (SAPMart Potsdam), New Status (New), Plan (13.10.2016 - 19.10.2016), Pre-dip, and Scan. The 'Tactics' section includes a 'Select Tactics' dropdown and a table for 'Media Support' and 'Display' options. Extension points are highlighted in yellow: 'extDetailedPromoGenData' and 'extDetailedPromoDateFields'. The right sidebar shows a summary for 'Plan: SAP Mart Norway North 2016' with Sales Volume (-134.89) and ROI (-102.02%).

View Extension Points for Detailed Promotion

3.3.6.1 Detailed Promotion Header



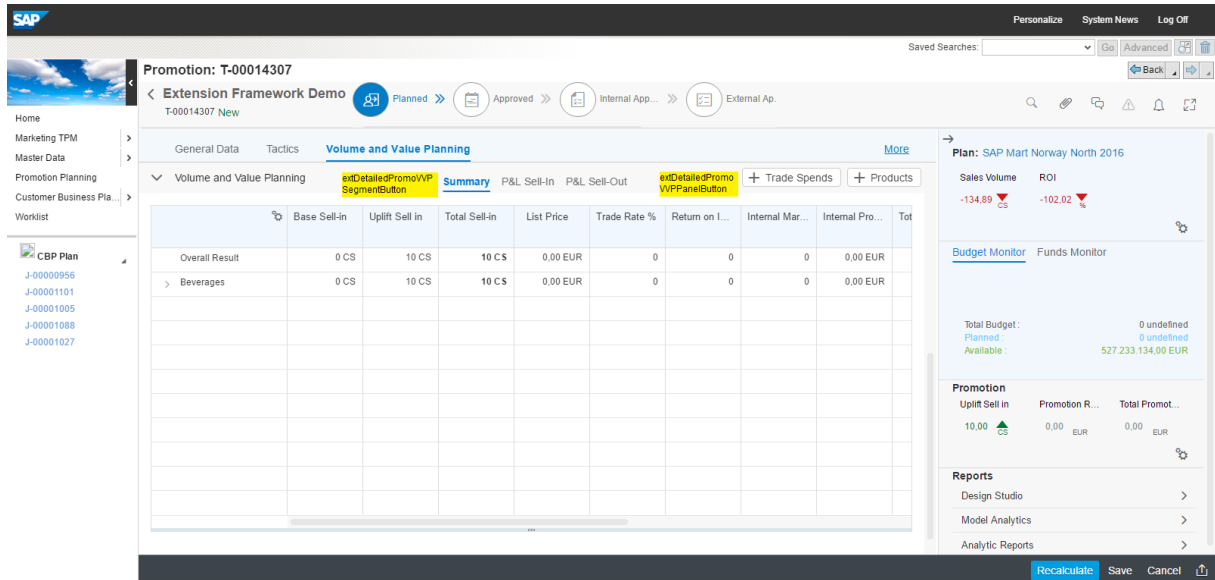
View Extension Points for Detailed Promotion Header

3.3.6.2 Footer Bar



View Extension Points for Footer Bar

3.3.6.3 Volume Planning



View Extension Points for Volume Planning

3.3.6.4 Tactics

Available Tactics Select Tactics:

TPR Media Support Display Feature Feature Ad

Planned Tactics

TPR:

	Pricing Tactics	ACV	Action
<input type="radio"/>	0,00	100,00	X
<input type="radio"/>	0,00	100,00	X
<input type="radio"/>	0,00	100,00	X
<input type="radio"/>	0,00	100,00	X

Media Support

<input type="checkbox"/>	TV Commercial	ACV: 100,00	X
<input type="checkbox"/>	Radio Commercial	ACV: 100,00	X
<input type="checkbox"/>	Online Banner	ACV: 100,00	X

Display

<input type="checkbox"/>	4-way	ACV: 100,00	X
<input type="checkbox"/>	End cap	ACV: 100,00	X
<input type="checkbox"/>	Sidekick	ACV: 100,00	X

Feature

Feature Ad

OK

View Extension Points for Tactics

3.3.6.5 Trade Spends

Promotion: T-00014307

Extension Framework Demo

Planned Approved Internal App... External

General Data Tactics Volume and Value Planning

Customer: SAPMart Potsdam Promotion Type: Detailed Promotion

Plan: 13.10.2016 - 13.10.2016

TV Commercial Radio Commercial Online B...

Volume and Value Planning

Trade Spends

Lumpsum (Di...) + Lumpsum (F...) + OFF-% +

OFF-Rate + ON-% + ON-Rate +

No trade spends added

Apply Cancel

Plan: SAP Mart Potsdam Super 2016

Sales Volume ROI: -134,80 CS -102,02 %

Total Budget: 527.250.000,00 EUR

Promotion: 10,00 CS 0,00 EUR 0,00 EUR

View Extension Points for Trade Spends

3.3.7 Promotion Landing Page

3.3.7.1 Landing Page Header

The screenshot shows the SAP Promotion Landing Page Header. At the top, there is a search bar with the text "Search: Promotions" and a dropdown menu showing "SAP Mart Potsdam Super 2016". Below the search bar, there is a table with columns: Actions, ID/Description, Type, Customer, Status, Plan Start/End Date, Buying Start/End Date, Tactic, and Product. The table contains three rows of promotion data. The first row is highlighted in blue. The second row is highlighted in light blue. The third row is highlighted in light blue. The table is titled "100 Promotions" and "94 Planned" and "6 Track". There are also buttons for "Bulk Actions" and "Advanced".

Actions	ID/Description	Type	Customer	Status	Plan Start/End Date	Buying Start/End Date	Tactic	Product
<input type="checkbox"/>	T-00015903 No Description	Detailed Pro...	SAPMart Potsdam	New	28.10.2016 - 04.11.2016	28.10.2016 - 04.11.2016	TV Commercial 11 More	Gum Care toothb 1 More
<input type="checkbox"/>	T-00015667 No Description	Detailed Pro...	SAPMart Potsdam	New	24.10.2016 - 30.10.2016	24.10.2016 - 30.10.2016	End cap	
<input type="checkbox"/>	T-00015570 No Description	Detailed Pro...	SAPMart Potsdam	New	12.12.2016 - 18.12.2016	05.12.2016 - 11.12.2016	TV Commercial 11 More	Berry Blue Soft ...

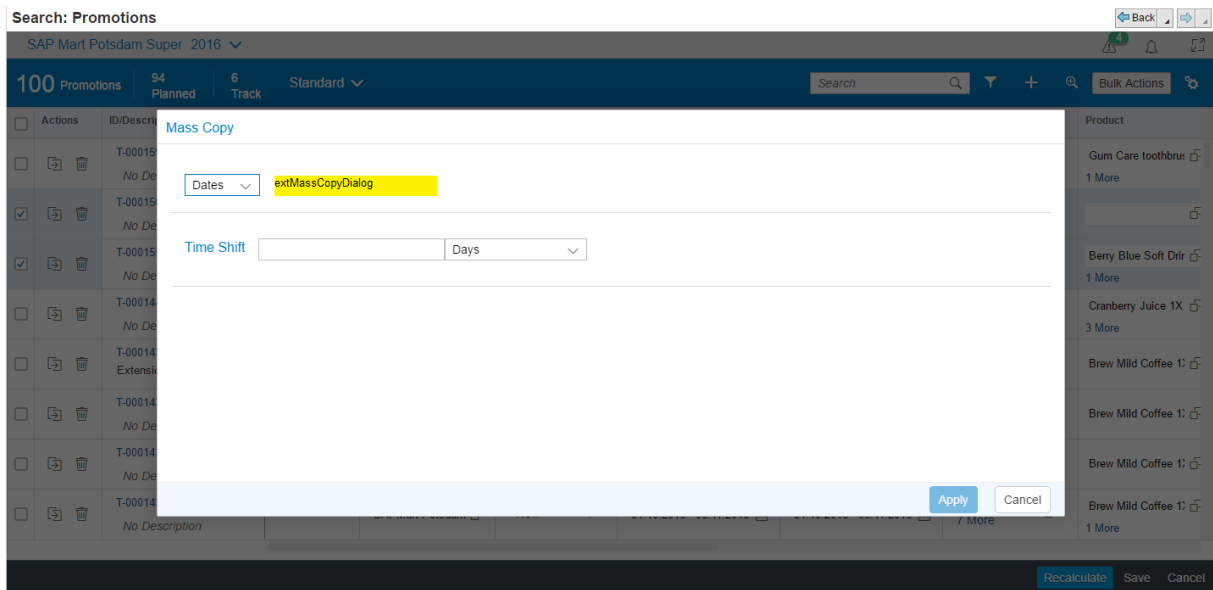
View Extension Points for Landing Page Header

3.3.7.2 Promotion Filter

The screenshot shows the SAP Promotion Filter. It features a grid of filter fields for various attributes. The fields are: Promotion ID, Promotion Type, Status, Employee Responsible, Product, Planning Account, Plan Start Date / End Date, Buying Start Date / End Date, Funds Plan, Sales Organization, Distribution Channel, and Division. Each field has a search icon. Below the filter fields, there is a yellow bar with the text "extFastPromotionFilter". At the bottom right, there is a "Max. Hits" field and buttons for "Apply", "Reset", and "Cancel".

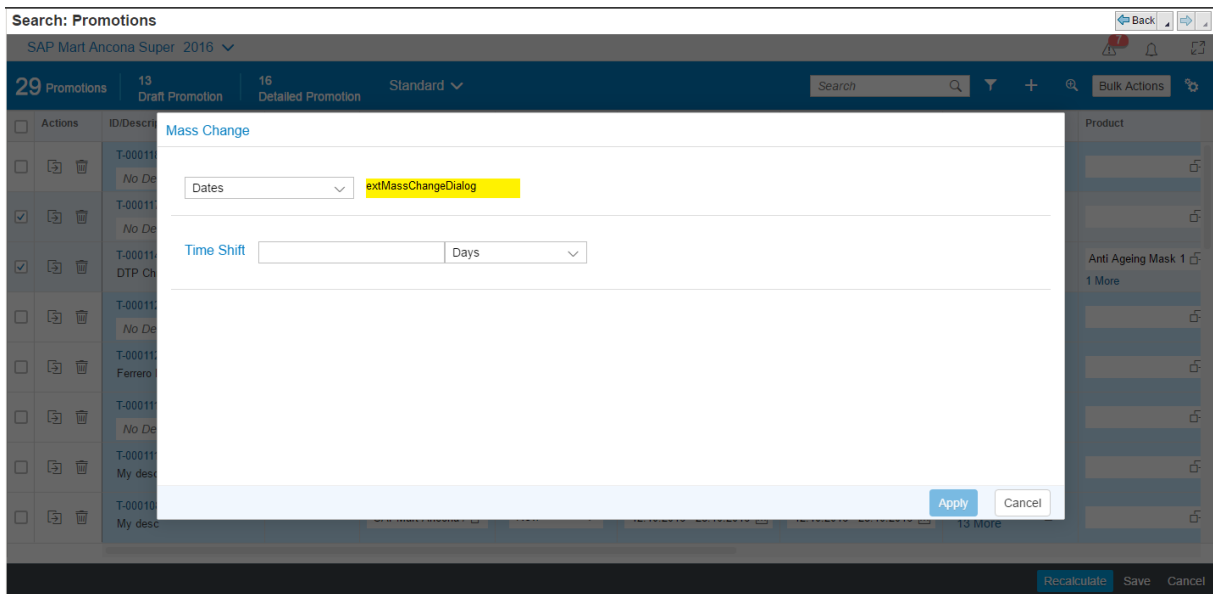
View Extension Points for Promotion Filter

3.3.7.3 Mass Copy Dialog



View Extension Points for Mass Copy Dialog

3.3.7.4 Mass Change Dialog



View Extension Points for Mass Change Dialog

3.3.8 Target Setting

The screenshot shows the 'Internal Target Planning' interface for 'SAP Mart Norway East 2016'. The main area is titled 'Target Setting (Volume)'. It features a navigation bar with 'Back', 'Forward', and 'Open' buttons. Below the navigation bar is a timeline for the year 2016, showing quarters and months. The main data area is a table with columns for 'Totals', 'Quarter1', 'Quarter2', 'Quarter3', and 'Quarter4', and rows for various factors like 'Economic Factor %', 'Corporate Factor %', etc. Extension points are highlighted in yellow: 'extIPTableToolBarLeft' and 'extIPTableToolBarRight' in the top right, and 'extFooterBarItem' in the bottom right.

View Extension Points for Target Setting

3.3.9 Target Setting Nav

The screenshot shows the 'Internal Target Planning' interface for 'SAP Mart Norway East 2016'. The main area is titled 'Target Setting (Volume)'. It features a navigation bar with 'Back', 'Forward', and 'Open' buttons. Below the navigation bar is a timeline for the year 2016, showing quarters and months. The main data area is a table with columns for 'Totals', 'Quarter1', 'Quarter2', 'Quarter3', and 'Quarter4', and rows for various factors like 'Economic Factor %', 'Corporate Factor %', etc. Extension points are highlighted in yellow: 'extHeaderRightItem', 'extSubToolBarRightItem', and 'extNavToolBarRightItem' in the top right, and 'extFooterBarItem' in the bottom right.

View Extension Points for Target Setting Nav

3.4 Controller Extension Table [extcontroller]

The Customizing activity for settings related to the controller extension table is [Extend UI Controller](#).

You can find the relevant extension information for your requirements in the following table.

Choose the *Controller Name* for the *Required Extension*.

For more information, see [Walkthrough – Controller Extension \[page 36\]](#).

Required Extension	Controller Name
Buyer Setup – Target Btn	sap.crm.mkt.cbp.lib.commons.controller.BuyerSetup
Scenario Planning – Toolbar Item	sap.crm.mkt.cbp.lib.commons.controller.ScenarioPlanning
Fast Promotions – Toolbar Item	sap.crm.mkt.cbp.lib.commons.controller.FastPromotions
Plan Setup -Secondary NavBar Tabs	sap.crm.mkt.cbp.lib.commons.controller.ScenarioPlanning
Common – Header Nav Bar Right Item	sap.crm.mkt.cbp.lib.commons.controller.Nav
Common – Header Top Bar Right Item	
Common – Header Top Bar Left Item	
Detailed Promotion Admin Area - Trade Promotions Details General Data	sap.crm.mkt.cbp.lib.commons.controller.AdminArea
Detailed Promotion Screen	sap.crm.mkt.cbp.lib.commons.controller.tma.DetailedPromotion
Volume Planning	sap.crm.mkt.cbp.lib.commons.controller.tma.VolumePlanning
Fast Promotions Filter	sap.crm.mkt.cbp.lib.commons.controller.FastPromotionsFilter
Tactics	sap.crm.mkt.cbp.lib.commons.controller.tma.Tactics
Fast Promotion	sap.crm.mkt.cbp.lib.commons.controller.FastPromotions

3.5 Post-Load Extension Table [extpostloadfiles]

The Customizing activity for settings related to the post-load extension table is [Maintain Post Load File](#).

You can find the relevant extension information for your requirements in the following table.

Choose the *Legacy Control* for the *Required Extension*.

For more information, see [Walkthrough – Post-Load Extension \[page 39\]](#).

Required Extension	Legacy Control
Plan – Toolbar Item	<code>sap.crm.mkt.cbp.lib.commons.control.PlanToolbar</code>
PPH – Action Item	<code>sap.crm.mkt.cbp.lib.commons.control.hierarchy.CustomHierarchy</code>
Assortment – Toolbar Item	<code>sap.crm.mkt.cbp.lib.commons.control.listing.ListingPanel</code>
ValueHelpMultiSelect – Add Button	<code>sap.crm.mkt.cbp.lib.commons.control.lib.valuehelp.ValueHelpMultiSelectInline</code>
ValueHelpMultiSelect – List Item Additional Info	
ValueHelpSingleSelect – Add Button	<code>sap.crm.mkt.cbp.lib.commons.control.lib.valuehelp.ValueHelpInline</code>
ValueHelpSingleSelect – List Item Additional Info	
RA Dropdown Menu	<code>sap.crm.mkt.cbp.lib.commons.control.PromoPlanningRespAreaDropDownMenu</code>

4 Walkthrough – View Extension

Prerequisites

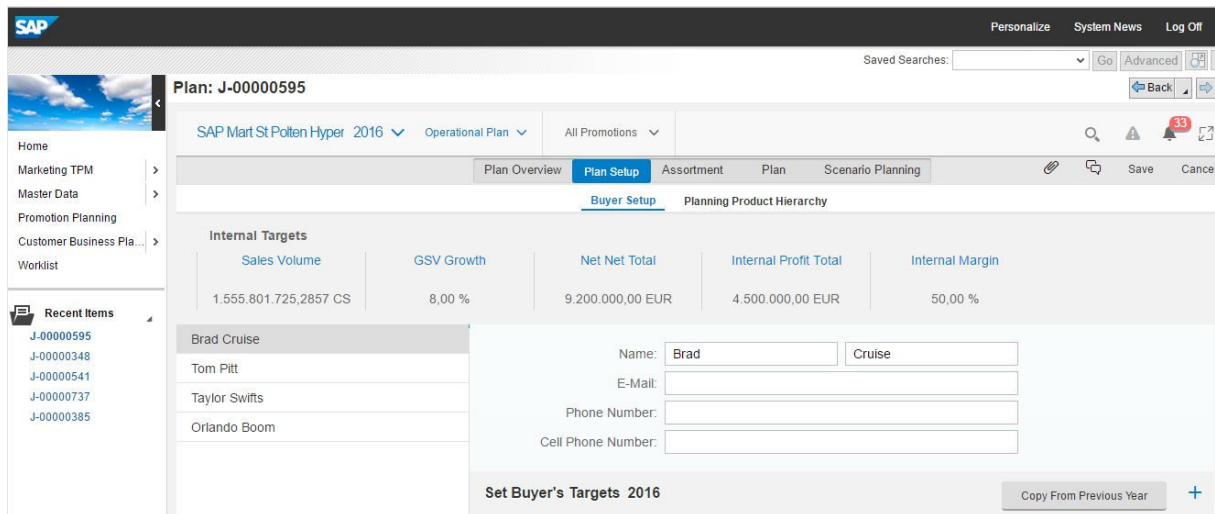
You have the relevant extension information from the extension table listed under [Extension Tables \[page 19\]](#). Refer to the [View Extension Table \[extviewextpoints\] \[page 19\]](#) to find the extensible areas of the *Buyer Setup* screen.

Context

Let's assume you want to extend the *Buyer Setup* view by adding a button to the *Buyer's Targets* toolbar.

Original Screen/Behavior

When you visit the *Buyer Setup* screen, the form looks as shown in the following figure:



Original Buyer Setup Screen

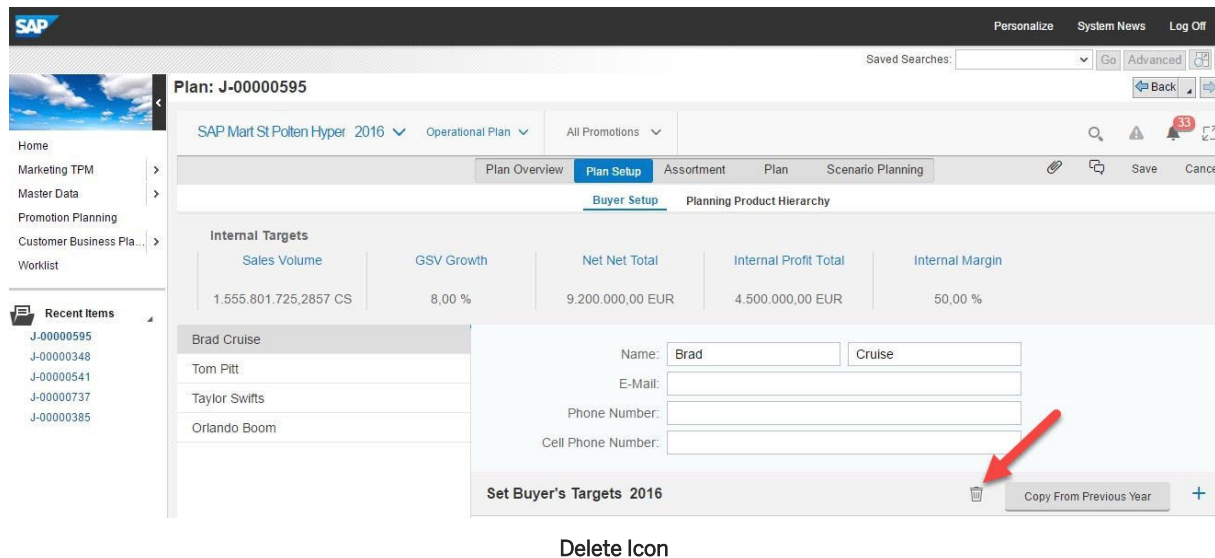
Purpose and Scope of the Extension

The purpose of this extension point is to add a

 *Delete* icon to the *Buyer's Targets* toolbar.

Expected Screen/Behavior

After adding the button, the *Buyer Setup* screen looks as shown in the following figure:



Delete Icon

Procedure

1. Identify the type of extension.

In the view extension table, find the exact extension point that matches the desired extension.

Note

If you cannot find the relevant details in the view extension table, these may be available in the post-load table ([Post-Load Extension Table \[extpostloadfiles\] \[page 30\]](#)).

If the extension information is neither available in the view extension table nor in the post-load table, the type of extension that you are trying to implement is not possible at this point.

You can use the application custom theme designer to hide certain controls or apply additional styling. You can access the custom theme designer at: https://<fully-qualified-host-name>:<port-number>/sap/bc/ui5_ui5/jbpc/ui5_thmdsg/index.html.

2. Identify the extension point.

Get the appropriate extension point from the view extension table. If the relevant information cannot be found, the extension is not possible using this approach.

The relevant information for this extension:

Required Extension	View Name	Extension Point Name
Buyer Setup – Target Btn	sap.crm.mkt.cbplib.commons.v iew.BuyerSetup	extBuyerSetupBuyerTargetBtn

3. Create the extended fragment file.

You can create a new view or fragment definition file that contains the desired structure you want to see at the extension point. You can store the extension files in a common folder, which will be deployed later to the ABAP Repository.

Example

The extended fragment file looks similar to the following example:

Extended Fragment File

4. Deploy the extended files to the ABAP Repository.

After making the changes, you can deploy/submit the extended files to the ABAP Repository. For more information, see [Deploying the Application as a BSP Application \[page 7\]](#).

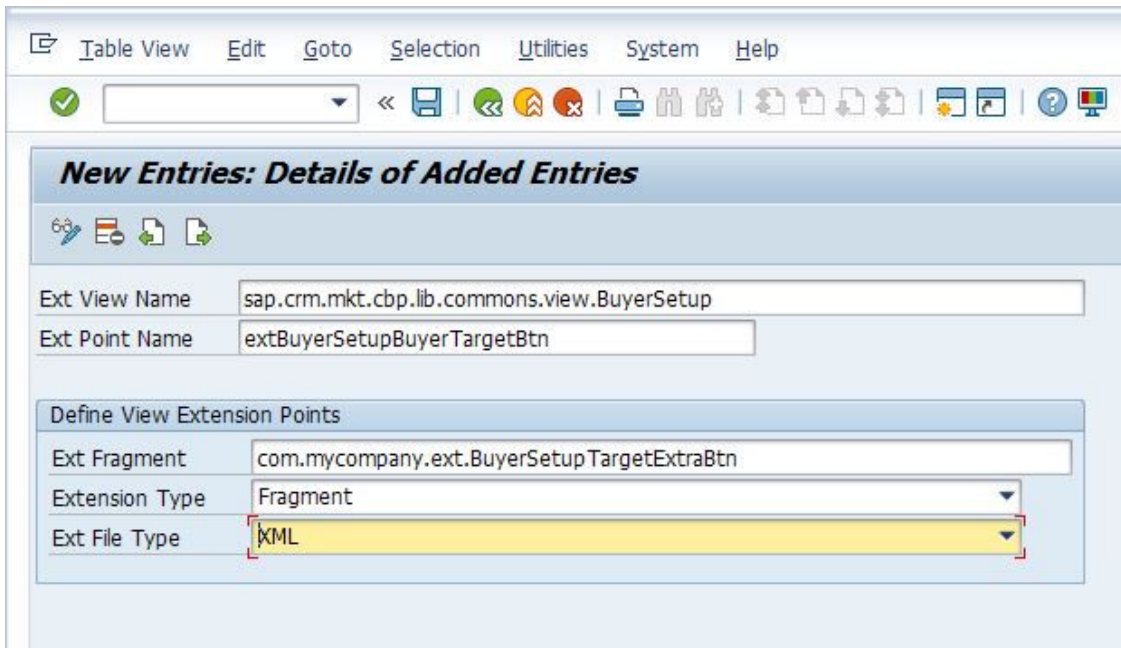
5. Maintain the Customizing settings.

Once the extended application has been deployed to the ABAP Repository, you can maintain the Customizing settings.

⚠ Caution

You must configure the settings for the namespace before you define the view settings, as otherwise the application may crash because the dependencies cannot be resolved.

- a. When configuring the settings, copy the exact view name and extension point from the view extension table.
- b. For the extended file name, copy only the extended file name without the type (.view/ .fragment) and file extension (.xml/ .js) parts. The API methods will automatically add these when loading.
- c. In the Customizing activity [Define View Extension Points](#), add a new entry as shown in the following figure:



Define View Extension Points

5 Walkthrough – Controller Extension


Prerequisites

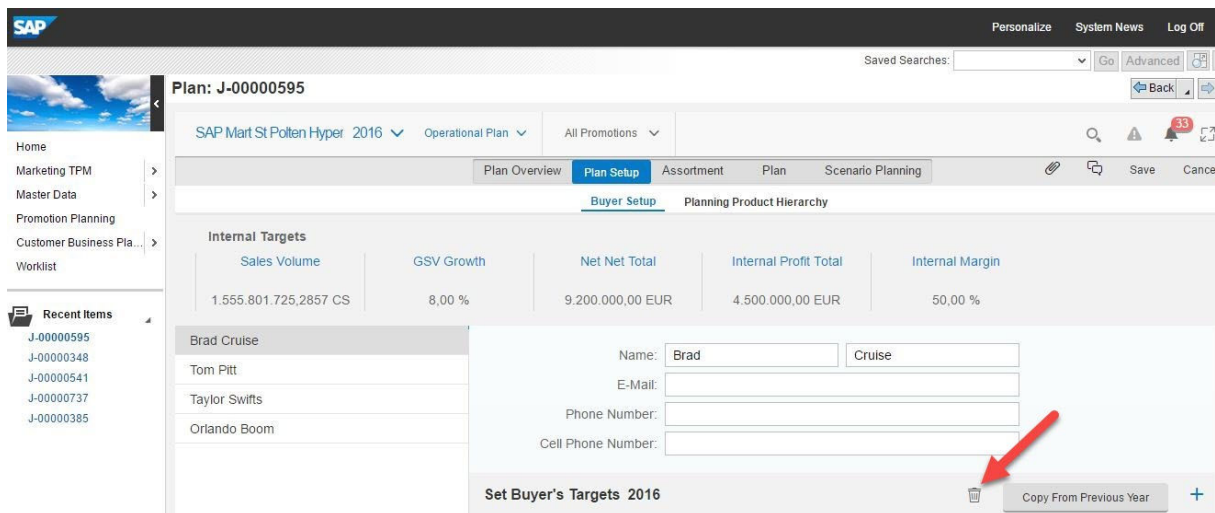
You have the relevant extension information from the extension table listed under [Extension Tables \[page 19\]](#). Refer to the [Controller Extension Table \[extcontroller\] \[page 29\]](#) to find the desired controller name.

Context

Let's assume you want to add a handler method attached to the control you created in [Walkthrough – View Extension \[page 32\]](#).

Original Screen/Behavior

In the previous walkthrough (view extension), we extended the *Buyer Setup* view and included the  *Delete* button to the *Buyer's Targets* toolbar.



Delete Icon

The new button, however, does not perform any actions, nor does it have a handler.

Purpose and Scope of the Extension

You might want to add the handler method in the extended controller file and bind this handler method to the press event of the button you created earlier.

i Note

The controller extension enables you to add new handlers and offers reusable methods, such as filter functions, and the ability to override existing methods.

Expected Screen/Behavior

After adding the controller extension, the new button has a handler and it can be used to perform an action.

Procedure

1. Identify the type of extension.

In the controller extension table, find the exact controller that matches the desired extension.

Note

If you cannot find the relevant details in the controller extension table, these may be available in the post-load table ([Post-Load Extension Table \[extpostloadfiles\]](#) [page 30]).

If the extension information is neither available in the controller extension table nor in the post-load table, the type of extension that you are trying to implement is not possible at this point.

2. Identify the controller.

Get the appropriate controller name from the controller extension table. If the relevant information cannot be found, the extension is not possible using this approach.

The relevant information for this extension is:

Required Extension	Controller Name
Buyer Setup – Target Btn	sap.crm.mkt.cbp.lib.commons.controller.Buyer Setup

3. Create the extended controller file.

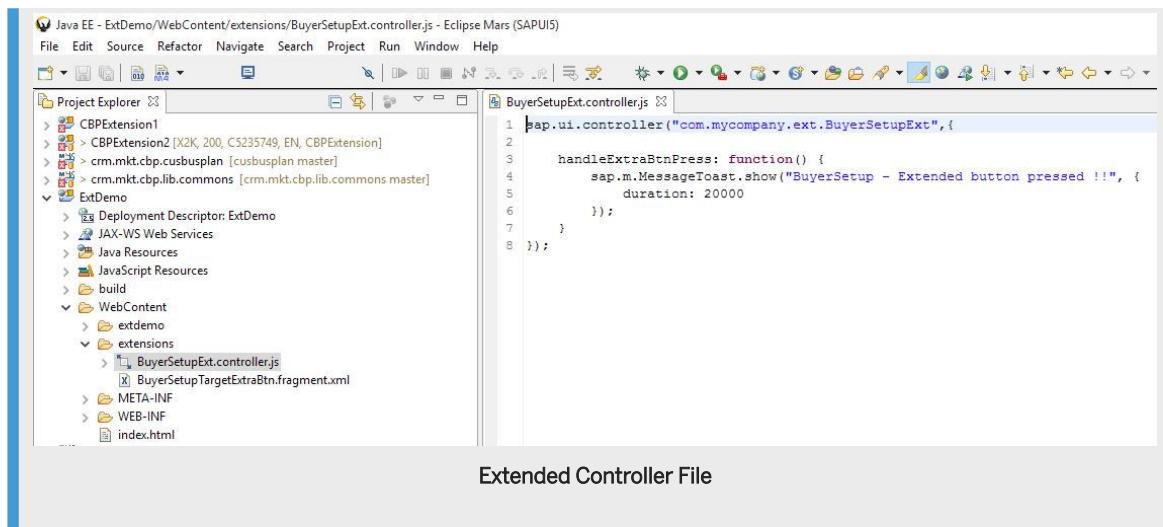
You can create a new controller file that contains the desired method implementation and overridden methods. You can store the extension files in a common folder, which will be deployed later to the ABAP Repository.

4. Implement the methods.

You can now add the handler methods or choose to override the existing methods. You can also write supporting or reusable methods inside the extended controller, such as filters.

Example

The extended controller file looks similar to the following example:



Extended Controller File

5. Deploy the extended files to the ABAP Repository.

After making the changes, you can deploy/submit the extended files to the ABAP Repository. For more information, see [Deploying the Application as a BSP Application \[page 7\]](#).

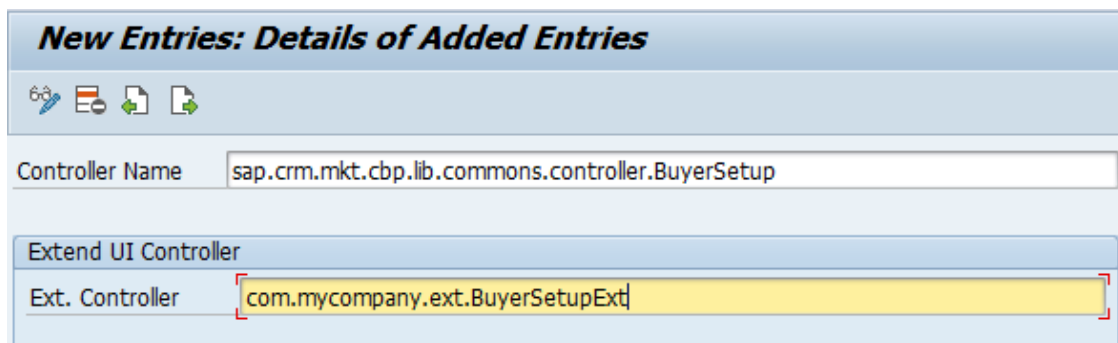
6. Maintain the Customizing settings.

Once the extended application has been deployed to the ABAP Repository, you can maintain the Customizing settings.

⚠ Caution

You must configure the settings for the namespace before you define the view settings, as otherwise the application may crash because the dependencies cannot be resolved.

- a. When configuring the settings, copy the exact controller name from the controller extension table.
- b. For extended file name, copy only the extended file name only, without the type (. controller) and file extension (. js) parts. The API methods will automatically add these when loading.
- c. In the Customizing activity *Extend UI Controller*, add a new entry as shown in the following figure:



Details of Added Entries

6 Walkthrough – Post-Load Extension

Context

In the application, some screens have a lot of controls that are rendered in such a way that they appear as views but are not actually views. Whenever we navigate away from those screens, some sets of controls are destroyed and recreated to form a view. In fact, all of them are recreated inside `Nav . view`.

Some parts of `Nav . view` that are always present during the life of the view can be used for view extensions using extension points. We have identified these areas and provided extension points for them.

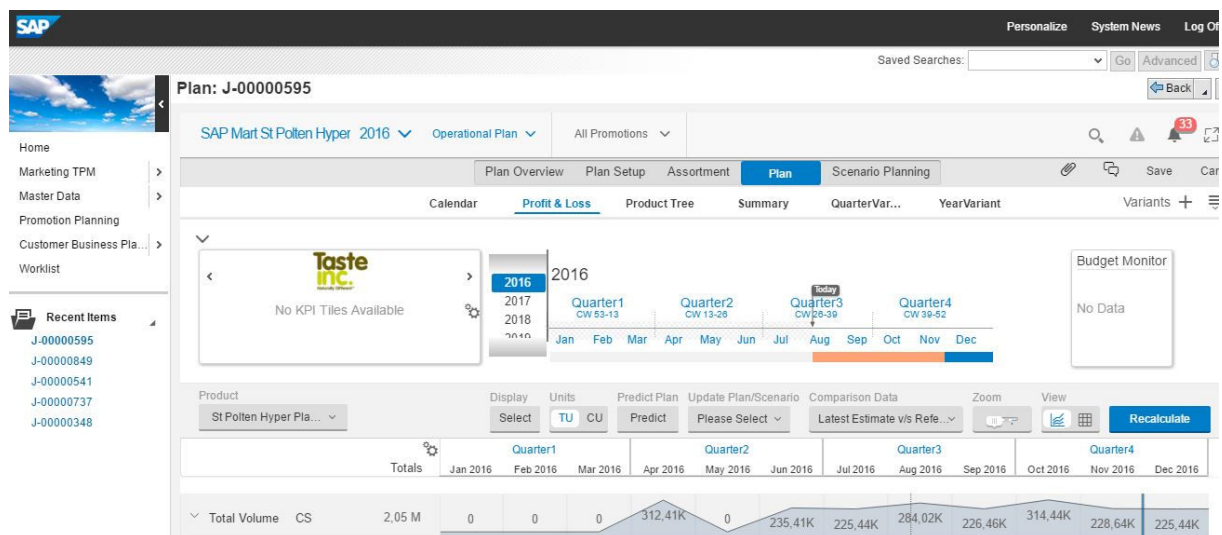
For other parts where the application uses legacy controls, however, the extension framework does not support the view and controller extensions. You can use a post-load extension to enhance these legacy controls.

Note

Post-load extensions enable you to override the predefined methods and implement a new logic to fulfill user requirements that define how the control looks and what action is performed when an event is triggered.

Original Screen/Behavior

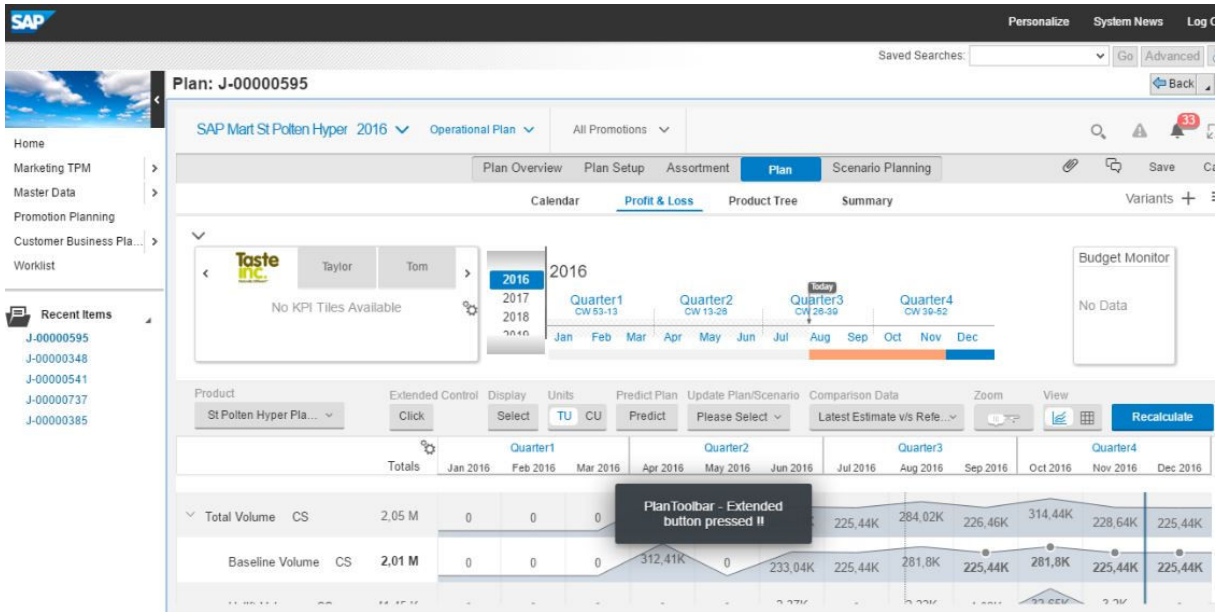
The *Plan* screen gives the feeling of an independent view. As explained, however, it is merely a set of controls rendered inside `Nav . view` that make it look like an independent view.



Plan Screen

Purpose and Scope of the Extension

The *Plan* screen has a toolbar containing various dropdowns and buttons, such as the *Recalculate* button. Let's consider a scenario where you would like to add a dropdown or a button to this toolbar.



Add Button to Toolbar

Procedure

1. Identify the type of extension.

In the post-load extension table, find the exact post-load information that matches the desired extension.

Note

If you cannot find the relevant details in the post-load extension table, these may be available in the view or controller extension table.

If the extension information is not available in the post-load extension table, controller extension table, or view extension table, the type of extension that you are trying to implement is not possible at this point.

2. Identify the post-load file name.

Get the appropriate post-load file name from the post-load extension table. If the relevant information cannot be found, the extension is not possible using this approach.

The relevant information for this extension is:

Required Extension	Legacy Control
Plan – Toolbar Item	sap.crm.mkt.cbp.lib.commons.control.PlanToolbar

3. Create the extended post-load file.

You can create a new post-load file, in which you enter a `require` statement and implement methods later. The `require` statement resolves all the dependencies for the legacy control.

You can add new methods or override the control prototype. You can store the extension files in a common folder, which will be deployed later to the ABAP Repository.

4. Implement the desired changes.

You can now add the new methods or choose to override the existing methods in the legacy control prototype. You can also write supporting or reusable methods inside the extended file, such as filters functions.

Example

The extended post-load file looks similar to the following example:

```

1 /*global sapIslandLib*/
2 (function() {
3     "use strict";
4     jQuery.sap.require("sap.crm.mkt.cb.lib.commons.control.PlanToolBar");
5
6     sap.crm.mkt.cb.lib.commons.control.PlanToolBar.prototype._addExtendedControl = function() {
7
8         // Add the display filter
9         var oDisplayFilter = this.displayFilter = new sap.crm.mkt.cb.lib.commons.control.DisplayFilter({
10             label: "Extended Control"
11         });
12
13         oDisplayFilter.addControl(new sap.ui.commons.Button({text: "Click", press: function() {
14             sap.m.MessageToast.show("PlanToolBar - Extended button pressed !!", {
15                 duration: 20000
16             }));
17         }));
18         this.addRightItem(oDisplayFilter);
19     };
20
21     sap.crm.mkt.cb.lib.commons.control.PlanToolBar.prototype._initToolBarContent = function() {
22
23         // Add extended control
24         this._addExtendedControl();
25     };
26 }());

```

Extended Post-Load File

5. Deploy the extended files to the ABAP Repository.

After making the changes, you can deploy/submit the extended files to the ABAP Repository. For more information, see [Deploying the Application as a BSP Application \[page 7\]](#).

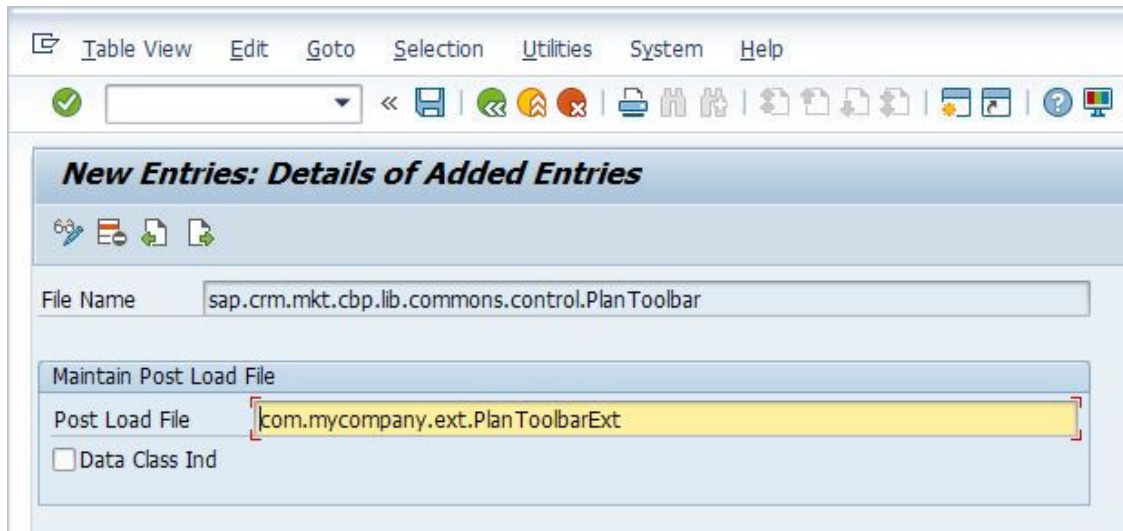
6. Maintain the Customizing settings.

Once the extended application has been deployed to the ABAP Repository, you can maintain the Customizing settings.

Caution

You must configure the settings for the namespace before you define the post-load settings, as otherwise the application may crash because the dependencies cannot be resolved.

- a. When configuring the settings, copy the exact controller name from the post-load extension table.
- b. In the Customizing activity *Maintain Post Load Files*, add a new entry as shown in the following figure:



Maintain Post Load Files

7 Walkthrough – Data Class Extension

Context

In the application, almost all the views and controls are bound with client-side UI5 models. Most of the controls and screens share common data models. These are named models and are made available with the SAPUI5 core object.

The application retrieves data from CRM and BW by means of service calls that can be represented in either XML or JSON format. It also receives a significant amount of information through the context nodes, which are arrays of objects containing key value pairs.

i Note

Since the data from various sources are in heterogeneous formats, they cannot be bound to SAPUI5 controls directly. We have to restructure them into a format that controls can understand and consume.

For the same purpose, we have data classes that have certain call-back methods, event handlers, and subscriptions to refresh the model data when necessary.

Original Screen/Behavior

The *Alert Inbox* shows all the alerts that have been generated. You can confirm these alerts by selecting some or all of them and clicking the *Mark as Complete* button.

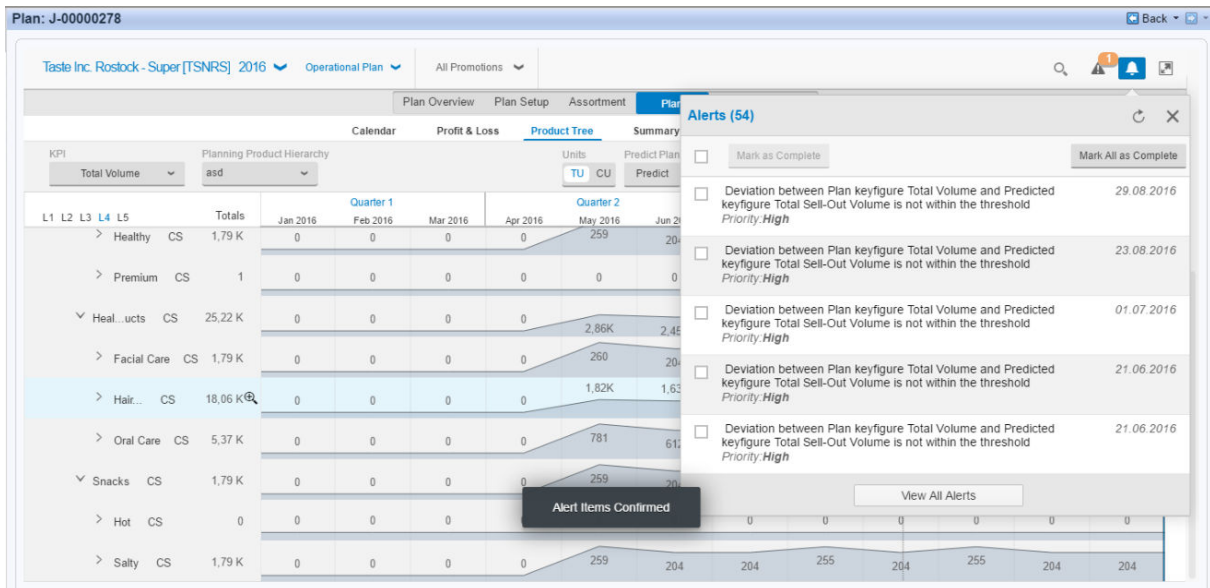
The screenshot shows the SAP Trade Management interface for 'Taste Inc. Roslock - Super [TSNRS] 2016'. The main window displays a dashboard with KPIs for Taste Inc. (GSV Growth, Cust. Margin, Net Net Total) and a calendar view. An 'Alerts (55)' popup is open on the right, listing several alerts with checkboxes and a 'Mark as Complete' button at the top right of the popup.

Mark as Complete Button

Purpose and Scope of the Extension

The *Alert Inbox* does not display a popup or notification when alerts are confirmed.

Let's consider a scenario where you would like to see a confirmation message after confirming alert items, as shown in the following figure:



All Items Confirmed

Procedure

1. Identify the type of extension.

Find the data class used for this extension. By inspecting the source, you can identify which data classes are used inside a view.

Note

You can patch certain methods or override the entire data class. The framework patches all the overridden methods and includes additional methods in the desired data class instance.

2. Identify the data class file name.

In the current example, we have the `AlertInboxData` directly, as we can check and analyze the `AlertInbox`-related files from the source.

The relevant information for this extension is:

Required Extension	Legacy Control
<code>AlertInboxData</code>	<code>sap.crm.mkt.cbp.lib.commons.data.AlertInboxData</code>

3. Create the extended file.

You can create an extended file for the data class, in which you enter a `require` statement and implement new methods or override existing ones later. The `require` statement resolves all the dependencies for the legacy control.

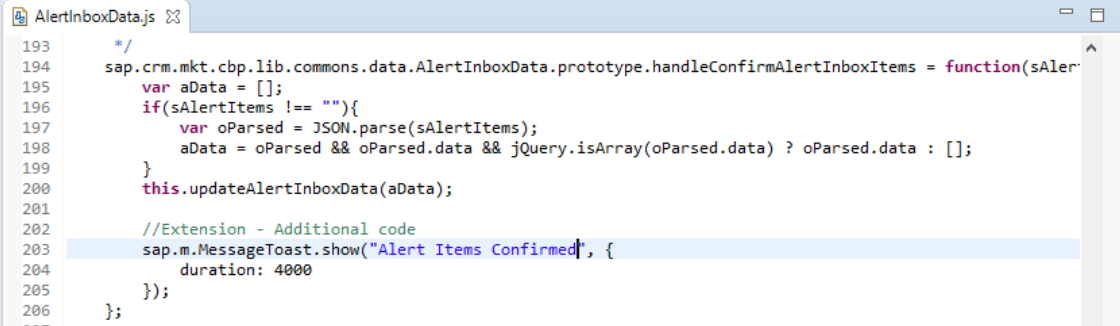
You can add new methods or override the control prototype. You can store the extension files in a common folder, which will be deployed later to the ABAP Repository.

4. Implement the desired changes.

You can now add the new methods or choose to override the existing methods in the original data class prototype. You can also write supporting or reusable methods inside the extended data class.

Example

The extended `AlertInboxData` file looks similar to the following example:



```

193  */
194  sap.crm.mkt.cbp.lib.commons.data.AlertInboxData.prototype.handleConfirmAlertInboxItems = function(sAlert
195  var aData = [];
196  if(sAlertItems !== ""){
197  var oParsed = JSON.parse(sAlertItems);
198  aData = oParsed && oParsed.data && jQuery.isArray(oParsed.data) ? oParsed.data : [];
199  }
200  this.updateAlertInboxData(aData);
201
202  //Extension - Additional code
203  sap.m.MessageToast.show("Alert Items Confirmed", {
204  duration: 4000
205  });
206  };
  
```

Extended AlertInboxData File

5. Deploy the extended files to the ABAP Repository.

After making the changes, you can deploy/submit the extended files to the ABAP Repository. For more information, see [Deploying the Application as a BSP Application \[page 7\]](#).

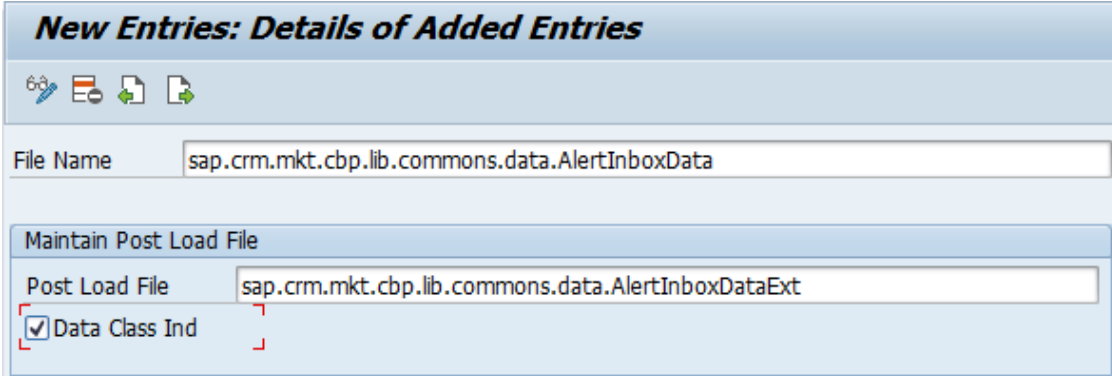
6. Maintain the Customizing settings.

Once the extended application has been deployed to the ABAP Repository, you can maintain the Customizing settings.

Caution

You must configure the settings for the namespace before you define the data class settings, as otherwise the application may crash because the dependencies cannot be resolved.

- When configuring the settings, copy the exact data class name identified for the data class extension.
- Select the *Data Class Indicator* check box.
- In the Customizing activity *Maintain Post Load Files*, add a new entry as shown in the following figure:



New Entries: Details of Added Entries

File Name:

Maintain Post Load File

Post Load File:

Data Class Ind

Data Class Indicator

8 Walkthrough – i18n Extension

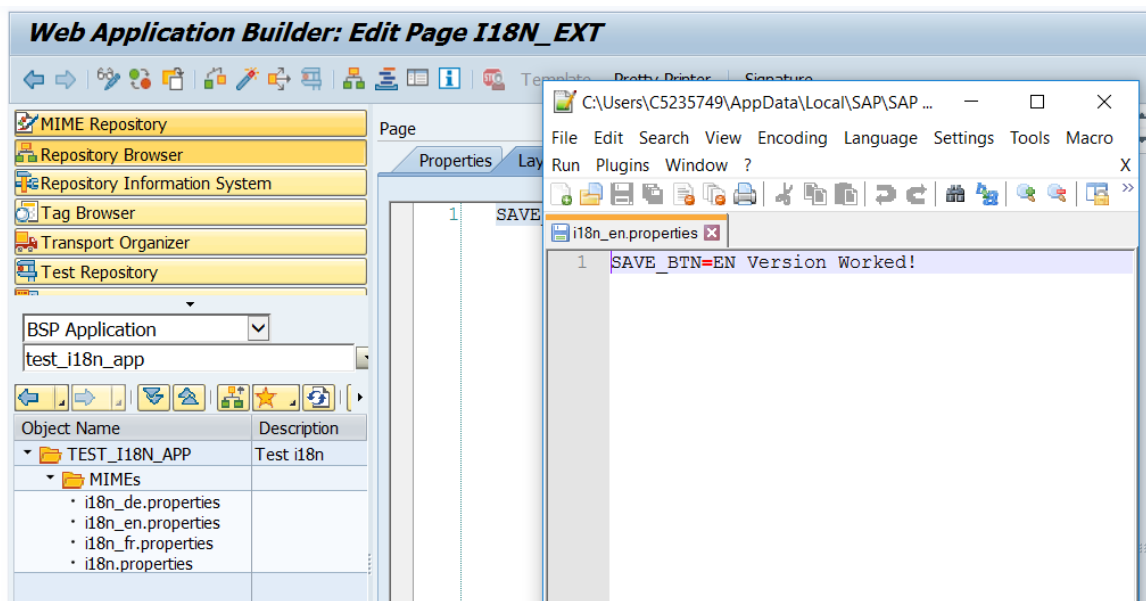
Procedure

1. Create a BSP application and add the required translation files.

You must add the custom i18n properties file to the MIME repository of a custom BSP application.

If the i18n texts are required in a different language, add the translated i18n properties file (for example, `i18n_de.properties`) to the same MIME repository folder.

If the text for any existing property needs to be changed, then provide the translated text for the same property present in the `i18n.properties` delivered by SAP.



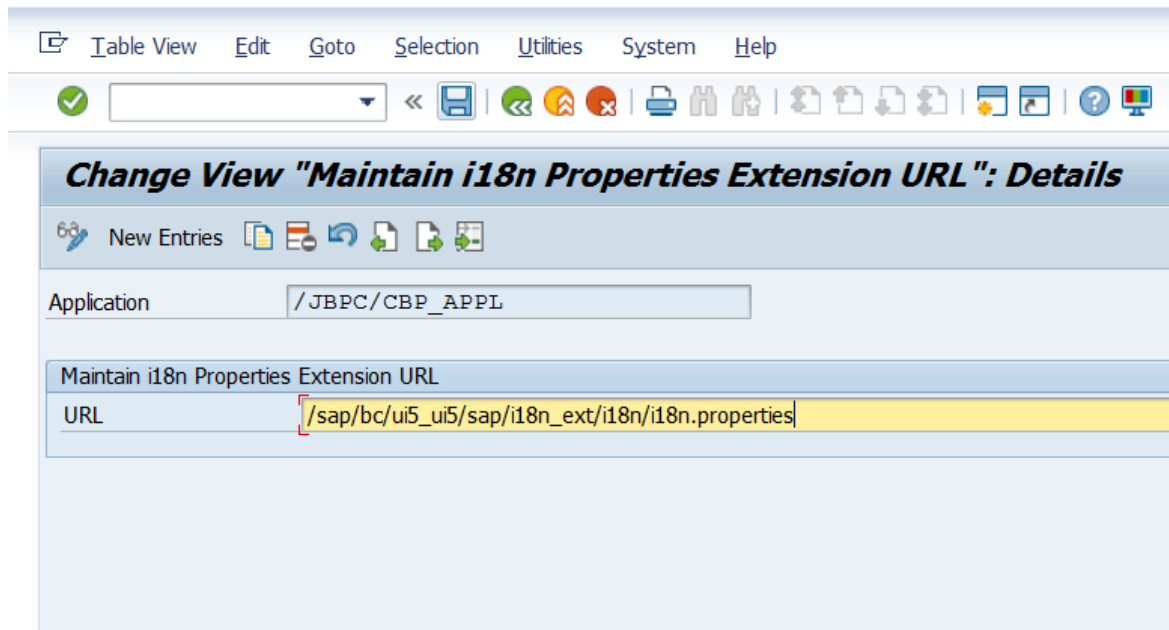
Change Property with Provided Text

2. Maintain the i18n properties extension URL in Customizing for *Customer Relationship Management* by choosing **Trade Management > General Settings > UI Configuration and Personalization > Maintain i18n Properties Extension URL**.

Maintain the URL of the custom i18n properties file that enhances the i18n properties file delivered with the application.

Example: Maintain i18n Properties Extension URL

Application Name	URL
/JBPC/CBP_APPL	/sap/bc/bsp/⟨⟨Z*BSP⟩⟩/i18n.properties
/JBPC/ITP_APPL	/sap/bc/bsp/⟨⟨Z*BSP⟩⟩/i18n.properties



Example: Maintain i18n Properties Extension URL

Results


The framework enhances the i18n model and loads the maintained texts for the specific language on the user interface.

Plan: J-00001187

SAP Mart North East 2017 | Operational Plan

Plan Overview | Plan Setup | Assortment | **Plan** | Scenario Planning

Calendar | Profit & Loss | Product Tree | **Summary** | Variants +



Nicole

No KPI Tiles Available

Internal Calendar

2017

2018

2019

Q012017 Q022017 Q032017 Q042017

CW52 - CW13 CW14 - CW26 CW27 - CW39 CW40 - CW52

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

TODAY

Budget Monitor

No Data

Planning Product Hierarchy: PPH1

Visible Rows: 25 | Visible Columns: 20 | Units: CU TU | Comparison Data: Latest Estimate vs ...

L1	L2	L3	L4	L5	CPV Price per Unit	Total Volume	Baseline	Impact	List Price	2 Gross External Sales	6a*PLD - Product Price List Discount on	6a* PLD - correction	6a* PLD - ECC	6a* PLD - (
Recalculate EN Version Worked! Cancel														



Language-specific Text Loaded on UI

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 an 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.

© 2023 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.