SAP NetWeaver
How-To Guide

How to enable Messaging to SAP UI5 Applications using ABAP Channels

Applicable Releases:

SAP NetWeaver AS ABAP 7.4 SP5

Target Audience:

ABAP and SAPUI5 Developers

Version 1.0

February 2014

Document History

Document Version	Description
1.00	First official release of this guide

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, F2 or ENTER.

Icons

Icon	Description
⚠	Caution
❗	Important
✎	Note
⬆	Recommendation or Tip
➔	Example

Table of Contents

1. Introduction 5

2. ABAP Channels – Overview 5

3. Why ABAP Push Channels? 5

4. Scenario Overview 6

5. Step-by-Step Procedure 6

5.1 Creating an SAP UI5 Application 6

5.1.1 Creating a Mobile Based SAP UI5 Application 7

5.1.2 Display Open Sales Orders by Dunning Level 9

5.1.3 Deploying the SAP UI5 Application to ABAP Application Server 11

5.1.4 Test the SAP UI5 Application 13

5.2 Creating ABAP Channels 14

5.2.1 Creating an ABAP Push Channel 14

5.2.2 Creating ABAP Messaging Channel 16

5.3 Enable APC to Notify the Changes (dunning level change) 17

5.3.1 Binding the ABAP Push Channel with ABAP Message Channel 17

5.3.2 Report to Modify the Dunning Level and Notify the Changes 18

5.3.3 Configure AMC to Listen and Notify the Dunning Level Change 18

5.4 Enable SAP UI5 Application to Receive the Notification from APC 20

5.4.1 Enable SAP UI5 Application to Listen to the Notification using Web Socket 20

5.4.2 Test the SAP UI5 Application with APC Notification 21

6. Appendix 22

6.1 Appendix A – Abbreviations 22

6.2 Appendix B – References 22

6.3 Appendix C – Link to ABAP for SAP HANA Reference Scenarios 22

6.4 Appendix D – Links to SAP NetWeaver Gateway Documents 22

1. Introduction

After reading this guide, you will be able to understand how to enable and configure ABAP Push channel (APC) to establish WebSocket based communication between two applications hosted on ABAP Application Server. This guide will assist you to:

- Create an SAP UI5 application (to view Open Sales orders by Dunning Level in Chart Control).
- Create an ABAP Push Channel and configure messaging channels.
- Enable APC for notifying the changes of dunning level to SAP UI5 application (during a change in dunning level).
- Enable SAP UI5 application to receive notifications from APC message channel.

2. ABAP Channels – Overview

ABAP Channels provide an event-based communication using messages between application servers and the Internet. ABAP channels technology enables efficient push-messaging between ABAP and UI sessions, including the respective user agents. This is available from SAP NetWeaver 7.4 SP05 onwards.

ABAP Channels mainly consists of two basic elements:

- A push channel, enabling bidirectional communication with user agents in the SAP NetWeaver Application Server.
- A publish/subscribe infrastructure (messaging channel) for the exchange of messages between different user sessions residing on different SAP NetWeaver Application Servers.

3. Why ABAP Push Channels?

The ABAP Push Channel (APC) provides a real-time Web behavior for collaborative scenarios in ABAP systems. It is based on an infrastructure to exchange messages between user contexts and UI sessions that reside on different application servers within the same system.

The diagram below shows message exchange between different sessions and user interfaces across different application servers.

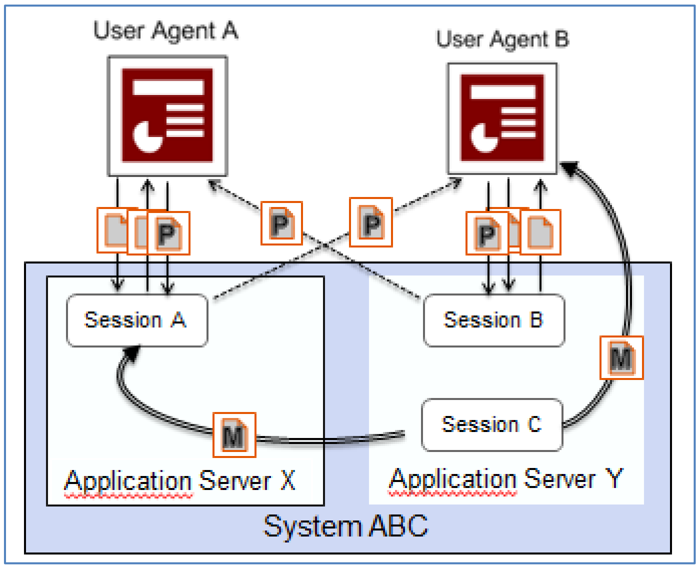


Figure 1: ABAP Push Channels

Typically, most of the ABAP applications use polling techniques to achieve an event-driven communication. For example, for pushing an event from ABAP backend to a browser based user-agent for example, WebDynpro, Business Server Pages (BSP) or Web GUI]; polling (in multi-seconds interval) against ICM cache is frequently used. This is a system resource consuming technique based on polling against a shared object area or database table, i.e. by checking the content of the shared area for the existence of an event. In SAP GUI, usually the timer control is used to detect an event in the backend.

By switching to push approach rather than the polling mechanism, as earlier, you can overcome some of the drawbacks. Moreover, with the possibility to host SAP UI5 [HTML5] applications in the ABAP Application Server, and with the support of Web Sockets in ABAP, it is now possible to develop very interactive and collaborative applications.

4. Scenario Overview

SAP UI5 application [as in this example] hosted on the ABAP Application Server consumes the underneath data sources (tables or views). There are various other mechanisms, which can consume and change the data sources such as:

1. Other transactional applications consuming the same data sources
2. Other users sessions of SAP UI5 application (if the UI5 application is transactional based)

Data gets invalidated when the resource is updated by other user agents. Hence the SAP UI5 application has to be notified (for all the users of all the sessions) to reload the data from the source. This document describes the step-by-step procedure of developing such a scenario enabling communication via notifications between different user agents.

This document will assist you to create a SAP UI5 application (mobile based) to view all the open sales orders grouped by dunning level, by consuming an ABAP Dictionary View ***SEPMAPPS_OPENINV***. In addition this document will also assist you to create an APC and configure a Messaging Channel to notify the change in dunning level. Finally the document contains information to adapt the SAP UI5 application to receive the notifications from APC via Web Sockets.

To achieve the use case described above, the following tasks have to be performed.

1. Create an SAP UI5 application and consume a data source (a HANA view).
2. Create an ABAP Push Channel service and configure Messaging channels.
3. Enable APC to notify the changes as and when the data source is updated.
4. Enable SAP UI5 application to communicate with APC.

*The SAP UI5 application uses a data source **SEPMAPPS_OPENINV,** which is an ABAP Dictionary View. The ABAP Dictionary view is the external view for the HANA Calculation View defined in HANA database. To avail the external view for hands-on, the underlying SAP NW release must be 7.40 SP02 or above.*

5. Step-by-Step Procedure

5.1 Creating an SAP UI5 Application

As a first step of our scenario, create an SAP UI5 mobile based application and consume the ABAP Dictionary view ***SEPMAPPS_OPENINV*** using a gateway service, which is already defined. To create an SAP UI5 application, you must complete the following tasks:

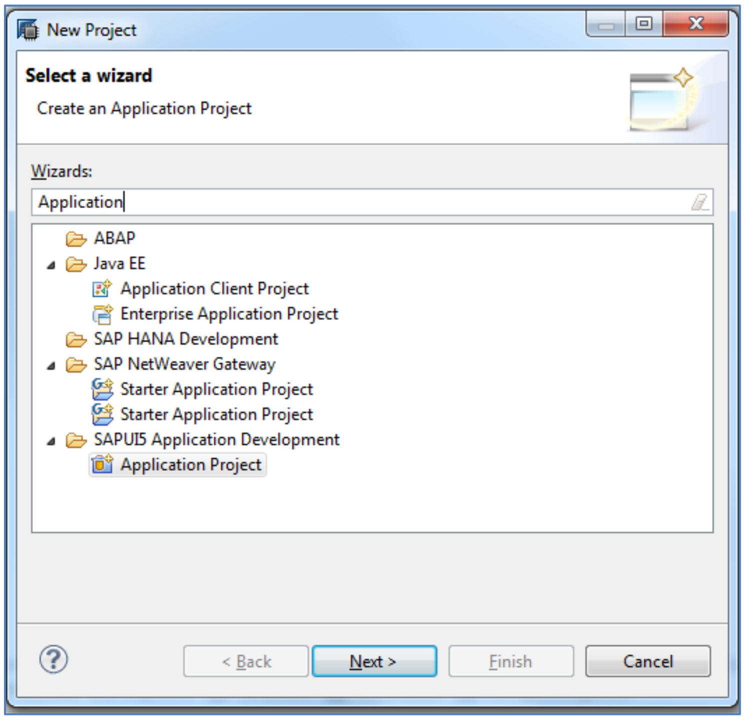
1. Create a mobile based SAP UI5 Application.
2. Display open sales orders by dunning level in UI5 chart control.

3. Deploy the SAP UI5 application.
4. Test the SAP UI5 application.

To create a SAP UI5 Application, the SAP UI5 Application plug-ins has to be installed on the Eclipse IDE.

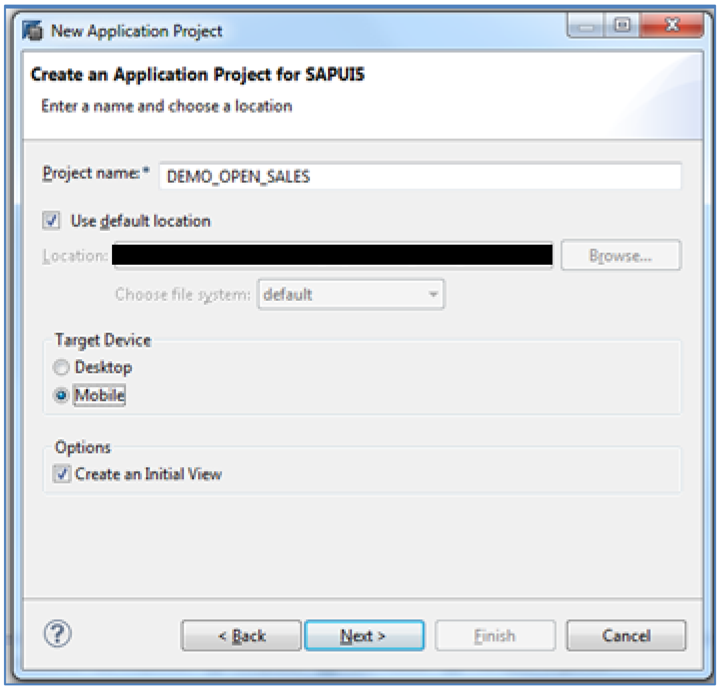
5.1.1 Creating a Mobile Based SAP UI5 Application

1. Launch the Eclipse IDE with the SAP UI5 tools.
2. Create a SAP UI5 application project. In the menu bar, navigate to ***File > New > Project***.
3. In the *New Project* wizard, type "***Application***" in filter.


New Project
Select a wizard
Create an Application Project
Wizards:
Application
ABAP
Java EE
Application Client Project
Enterprise Application Project
SAP HANA Development
SAP NetWeaver Gateway
Starter Application Project
Starter Application Project
SAPUI5 Application Development
Application Project
< Back
Next >
Finish
Cancel

4. Choose "**Application *Project***" under SAPUI5 Application Development.
5. Choose ***Next***.
6. Enter a name for the project. Example: ***DEMO_OPEN_SALES***.
7. Select Target Device as "***Mobile***".

Alternatively, you can also select the Desktop as Target Device for the sample application.



Create an Application Project for SAPUI5

Enter a name and choose a location

Project name:* DEMO_OPEN_SALES

[✓] Use default location

Location: [redacted]

Choose file system: default

Target Device

( ) Desktop

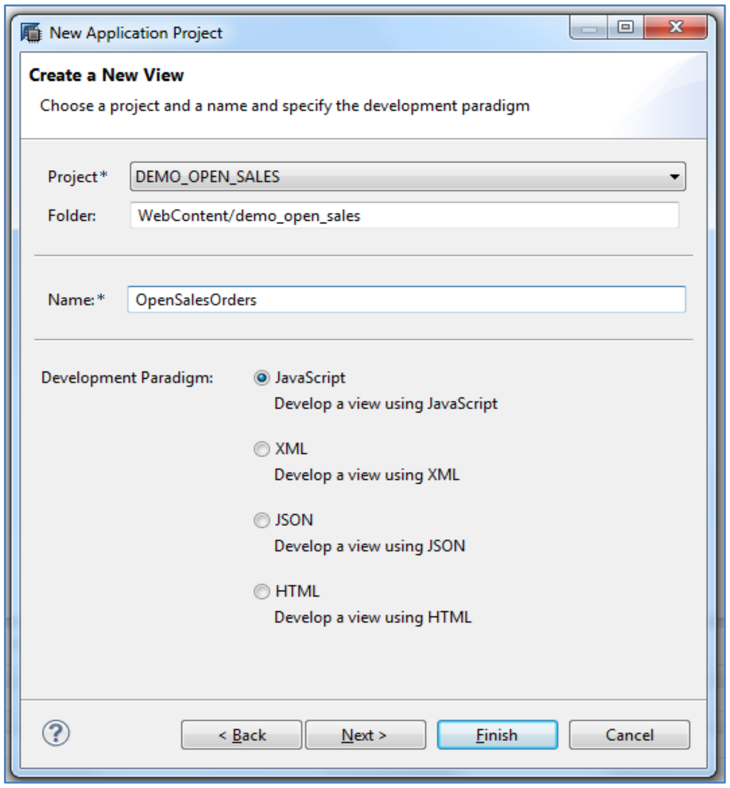
(●) Mobile

Options

[✓] Create an Initial View

8. Ensure that the "***Create an Initial View***" checkbox is selected.
9. Choose ***Next***.
10. In the next window, enter a name for view. Example: ***OpenSalesOrders***.

With this name, the View and the Controller files are created after completion.



Create a New View

Choose a project and a name and specify the development paradigm

Project* DEMO_OPEN_SALES

Folder: WebContent/demo_open_sales

Name:* OpenSalesOrders

Development Paradigm:

(●) JavaScript
Develop a view using JavaScript

( ) XML
Develop a view using XML

( ) JSON
Develop a view using JSON

( ) HTML
Develop a view using HTML

11. Retain Development Paradigm as “***JavaScript***”.
12. Choose ***Next***. A summary page is displayed.
13. Choose ***Finish***.

   The project appears in the *Project Explorer* view.

5.1.2 Display Open Sales Orders by Dunning Level

The steps in section Creating a Mobile Based SAP UI5 Application creates a skeleton of the SAP UI5 application.

The following steps explain how to embed a chart control (from SAP UI5 Viz Library) to display open sales orders from the data source ***SEPMAPPS_OPENINV,*** using OData model binding. This involves the following steps:

1. Create a new OData model object.
2. Embed the chart control to display open sales orders by dunning level.

5.1.2.1 Creating a New OData Model Object

In the controller file (***OpenSalesOrders.controller.js***), there are predefined lifecycle hooks that you can implement. The ***onInit*** function method is called when its corresponding View is instantiated and its controls (if available) are already created. This method can also be used to do one-time initialization. Ideally, it is in this method the OData model should be initialized. The ***OData Model*** enables binding of controls to data from OData Service.

The Gateway service has been created to consume the data source SEPMAPPS_OPENINV and expose the data in OData format.

The implementation of the onInit method is as below. Copy and paste the same in the ***OpenSalesOrders.controller.js*** *file.* This will create an OData model and this model refers to the service URL "/sap/opu/odata/sap/EPM_OIA_APPS_GW_SERVICE_SRV/".

```
onInit: function() {
    var oDunningAnalysisModel;
    //initializaing odata model with gateway odata service url
    oDunningAnalysisModel = new   sap.ui.model.odata.ODataModel(
      "/sap/opu/odata/sap/EPM_OIA_APPS_GW_SERVICE_SRV/" );
    //as count is not needed for this app, disable the count functionality
    oDunningAnalysisModel.setCountSupported(false);
    //binding the oData model to core
    sap.ui.getCore().setModel(oDunningAnalysisModel);
},
```

5.1.2.2 Embed Chart control to display Open Sales Orders by Dunning Level

The OData Model is a server-side model, so the whole dataset is only available on the server. The data provided by the OData Model can be accessed according to the structure of the OData service. At this point, you can create controls, attach the OData Model and define the binding to the control properties in order to display data. To embed the chart and do the necessary data binding, copy the code snippet as given below in the view file (***OpenSalesOrders.view.js***) inside the ***createContent*** method.

```
createContent : function(oController) {
    // initialize chart dataset
    oDunningAnalysisChartDataset = new sap.viz.ui5.data.FlattenedDataset(
        "idDunningAnalysisChartDataset", {
        dimensions : [ {
            axis : 1,
            name : "Dunning Level",
            value : "{DunningLevel}"
         } ],
         measures : [ {
             name : "Sales Orders",
             value : "{NoSalesOrders}"
         } ],
         data : {
             path : "/DunningLevelDetails"
         }
    });
    // Initialize donut chart
    oDunningAnalysisChart = new sap.viz.ui5.Donut("idDunningAnalysisChart",
    {
        title : {
            visible : true,
            text : "Sales Orders by Dunning Level"
        },
        dataset : oDunningAnalysisChartDataset
    });
    // initialize page
    oDunningAnalysisPage = new sap.m.Page({
        title : "Dunning Analysis",
        content : [ oDunningAnalysisChart ] // add the chart to the page
    });
    // return the object from createContent function
    return oDunningAnalysisPage;
}
```

The above code snippet creates an instance of dataset by defining the dimension and measures for the chart control. Here the dunning level is the dimension and number of open sales orders is the measure. The OData model path also has been configured to load the open sales orders information from the gateway service. Then a Donut chart has been instantiated and the dataset has been associated to it. Finally, the chart is placed in the Page control.

5.1.2.3 The index.html

After creating the View and the Controller files, you need to provide the necessary details in the index.html. This is the entry page of the application. In the first script block below, the index.html file contains references to the libraries ***sap.m, sap.viz***, and the used theme ***sap_mvi***. In the second script block below, the index file creates a mobile application and an initial page and assigns the view to the initial page. Open the file ***index.html,*** copy and paste the code given below***.***

```
<script src="resources/sap-ui-core.js"
        id="sap-ui-bootstrap"
        data-sap-ui-libs="sap.m,sap.viz"
        data-sap-ui-theme="sap_mvi" >
</script>

<script>
       sap.ui.localResources("demo_open_sales");
```

```
        // create a mobile App
        // it initializes the HTML page for mobile use and
        // provides animated page handling
        var app = new sap.m.App({initialPage:"idOpenSalesOrders1"});
        var page = sap.ui.view({id:"idOpenSalesOrders1",
                         viewName:"demo_open_sales.OpenSalesOrders",
                         type:sap.ui.core.mvc.ViewType.JS});
        app.addPage(page);
        app.placeAt("content");
</script>
```

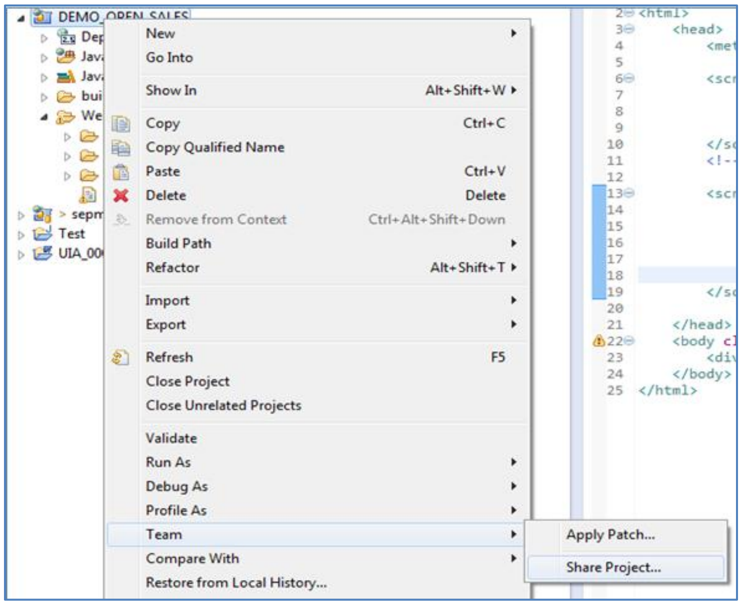
*The library **sap.viz** must be included for embedding the chart control.*

5.1.3 Deploying the SAP UI5 Application to ABAP Application Server

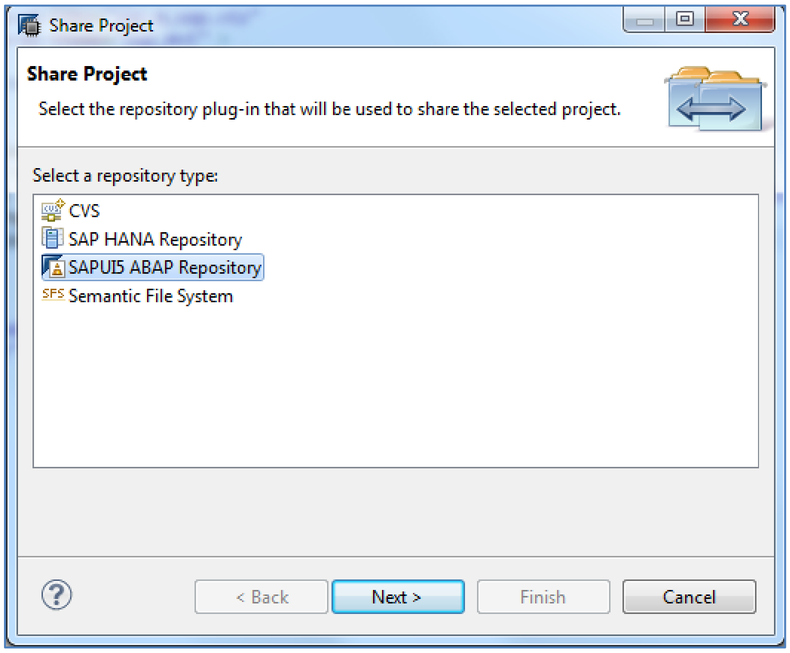
You need to deploy the SAP UI5 application on the ABAP Application server to run the application.

To deploy the SAP UI5 application, follow the steps below:

1. In the context menu of the SAP UI5 project, ***choose Team*** > ***Share Project.***



2. In the wizard, choose the repository type as ***SAPUI5 ABAP Repository.***
3. Choose ***Next***.



Share Project

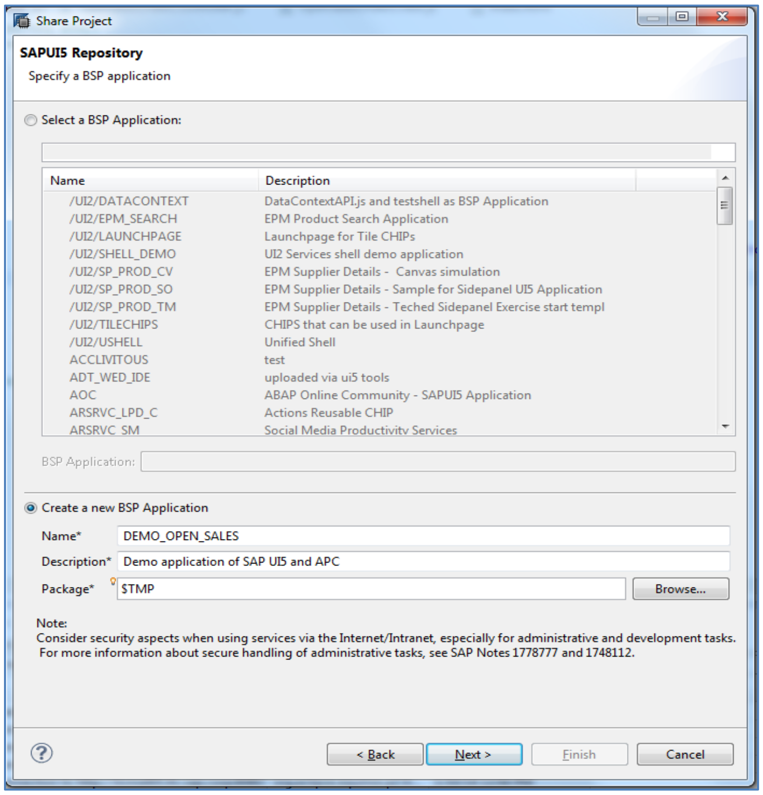
Share Project

Select the repository plug-in that will be used to share the selected project.

Select a repository type:

- CVS
- SAP HANA Repository
- SAPUI5 ABAP Repository
- Semantic File System

4. Choose the connection to your ***ABAP System.***
5. Choose ***Next***.
6. Logon to the ABAP system by providing the user credentials
7. Choose ***Next***.
8. Choose the option **Create a new BSP Application** as shown in the snapshot below**.**
9. Enter the ***name, description, and package*** of the BSP application.



Share Project

SAPUI5 Repository

Specify a BSP application

( ) Select a BSP Application:

Name	Description
/UI2/DATACONTEXT	DataContextAPI.js and testshell as BSP Application
/UI2/EPM_SEARCH	EPM Product Search Application
/UI2/LAUNCHPAGE	Launchpage for Tile CHIPs
/UI2/SHELL_DEMO	UI2 Services shell demo application
/UI2/SP_PROD_CV	EPM Supplier Details - Canvas simulation
/UI2/SP_PROD_SO	EPM Supplier Details - Sample for Sidepanel UI5 Application
/UI2/SP_PROD_TM	EPM Supplier Details - Teched Sidepanel Exercise start templ
/UI2/TILECHIPS	CHIPS that can be used in Launchpage
/UI2/USHELL	Unified Shell
ACCLIVITOUS	test
ADT_WED_IDE	uploaded via ui5 tools
AOC	ABAP Online Community - SAPUI5 Application
ARSRVC_LPD_C	Actions Reusable CHIP
ARSRVC_SM	Social Media Productivity Services

BSP Application:

[✓ Create a new BSP Application]

Name* DEMO_OPEN_SALES

Description* Demo application of SAP UI5 and APC

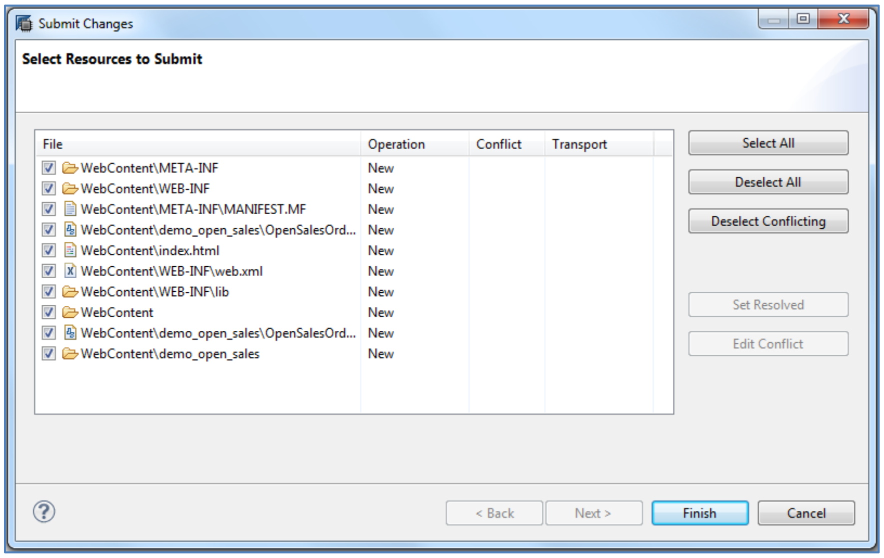
Package* $TMP

Note:
Consider security aspects when using services via the Internet/Intranet, especially for administrative and development tasks.
For more information about secure handling of administrative tasks, see SAP Notes 1778777 and 1748112.

10. Choose ***Next*** and provide the ***Transport Request*** number if required (other than $TMP package).
11. Choose ***Finish***.

A new BSP application is created in the ABAP Application Server. The next step is to submit the SAP UI5 code to ABAP (in the BSP application). Follow the steps below to submit the code to ABAP Repository:

1. In the context menu of the SAP UI5 project, choose ***Team*** > ***Submit***.


Submit Changes

Select Resources to Submit

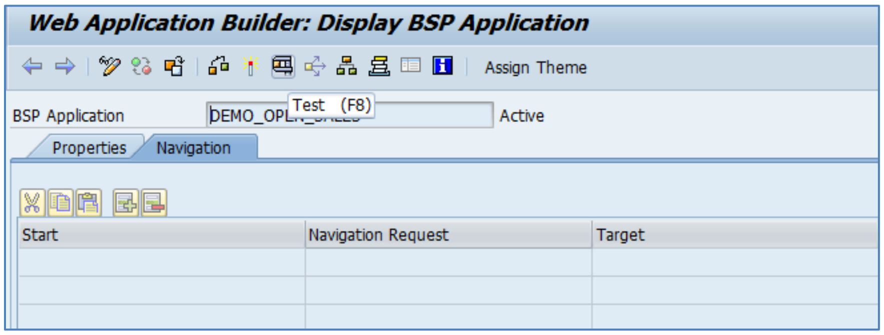
File	Operation	Conflict	Transport
WebContent\META-INF	New		
WebContent\WEB-INF	New		
WebContent\META-INF\MANIFEST.MF	New		
WebContent\demo_open_sales\OpenSalesOrd...	New		
WebContent\index.html	New		
WebContent\WEB-INF\web.xml	New		
WebContent\WEB-INF\lib	New		
WebContent	New		
WebContent\demo_open_sales\OpenSalesOrd...	New		
WebContent\demo_open_sales	New		

2. Choose Select All.
3. Choose Finish.

5.1.4 Test the SAP UI5 Application

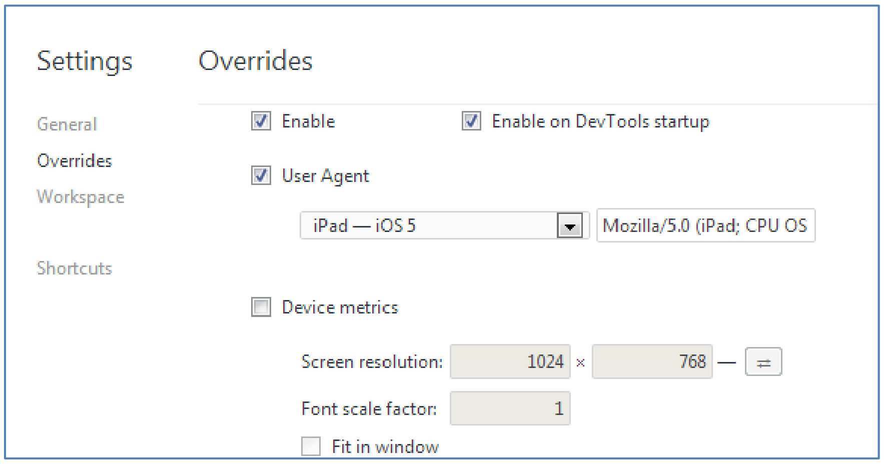
The SAP UI5 application is created and deployed on the ABAP Server. To test the SAP UI5 application, open the BSP application ***DEMO_OPEN_SALES*** and execute (**F8**) the application.

To open the BSP application, press `CTRL + SHIFT + A` *and enter the name* ***DEMO_OPEN_SALES*** *to navigate to the BSP application.*


Web Application Builder: Display BSP Application

BSP Application DEMO_OPEN_SALES Active

To view SAP UI5 application better, use the Chrome browser. Open Developer Tools (F12) and set the user agent as iPad or iPhone (Settings → Overrides → User Agent).



☑ Enable ☑ Enable on DevTools startup

☑ User Agent

iPad — iOS 5 | Mozilla/5.0 (iPad; CPU OS

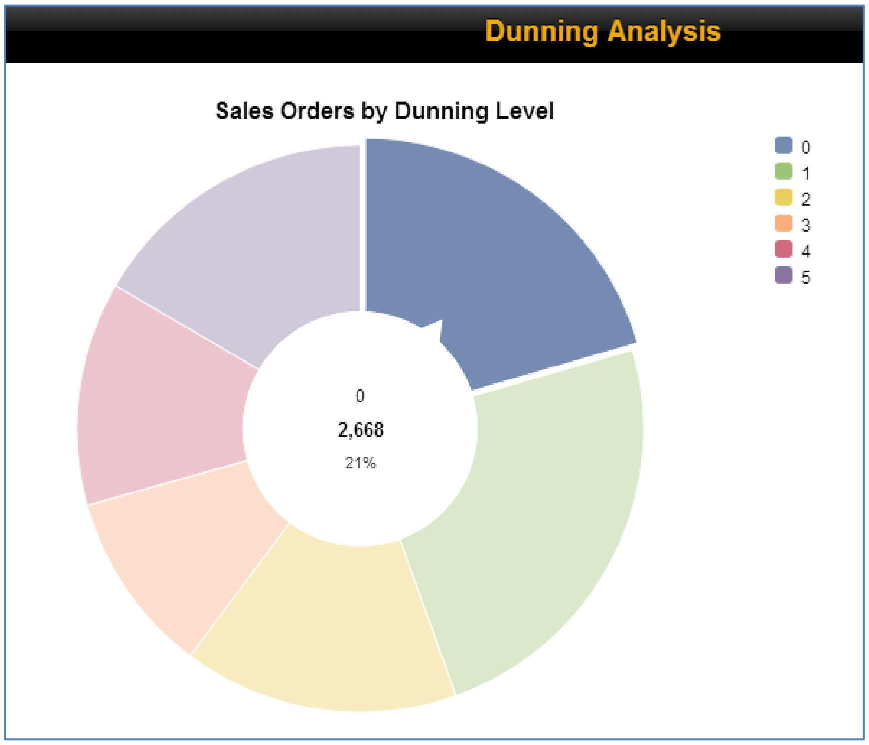
☐ Device metrics

Screen resolution: 1024 × 768

Font scale factor: 1

☐ Fit in window

The SAP UI5 application is loaded with a Donut chart displaying the open sales orders by dunning level as shown in the screenshot below.

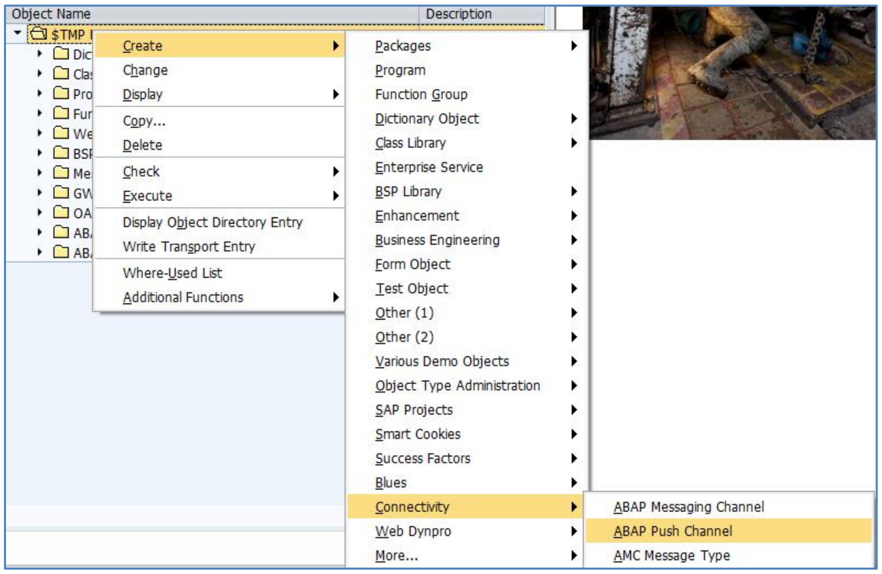


5.2 Creating ABAP Channels

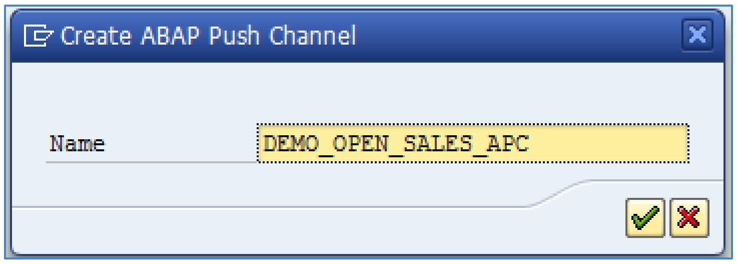
This section explains how to create an ABAP Push channel (APC) service and How to configure an ABAP Messaging Channel (AMC).

5.2.1 Creating an ABAP Push Channel

1. Logon to the ABAP system and open ABAP Workbench (SE80).
2. Choose the required ABAP package.
3. In the context menu, choose ***Create*** > ***Connectivity*** > ***ABAP Push Channel.***

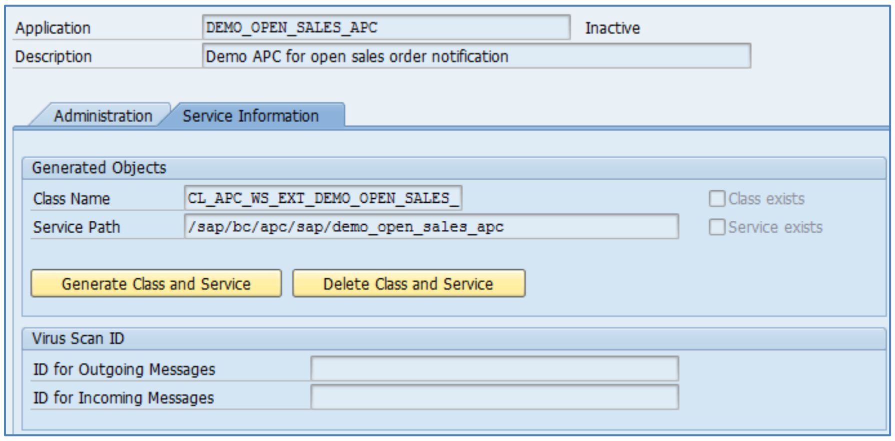


4. Enter the APC name as ***DEMO_OPEN_SALES_APC.***


Create ABAP Push Channel

Name DEMO_OPEN_SALES_APC

5. Enter the ***Package*** and ***Transport Request*** number (if required) details.
6. Enter the APC ***description.***
7. Save the APC.
8. Choose "***Generate Class and Service***".


Application DEMO_OPEN_SALES_APC Inactive
Description Demo APC for open sales order notification

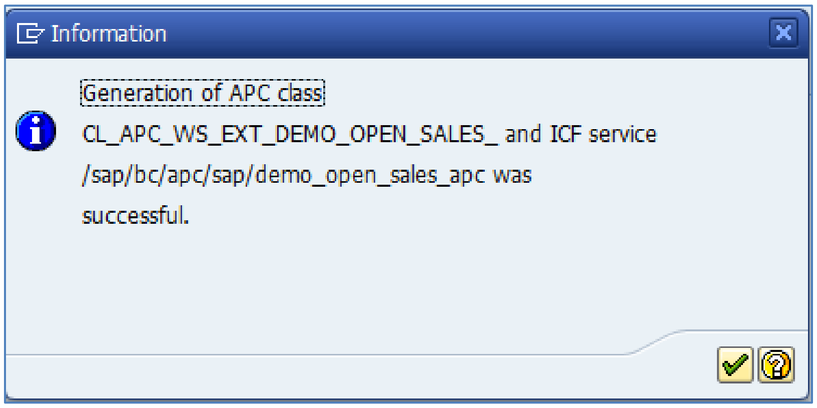
Administration | Service Information

Generated Objects
Class Name CL_APC_WS_EXT_DEMO_OPEN_SALES_ ☐ Class exists
Service Path /sap/bc/apc/sap/demo_open_sales_apc ☐ Service exists

Generate Class and Service | Delete Class and Service

Virus Scan ID
ID for Outgoing Messages
ID for Incoming Messages

9. A confirmation dialog displays the service end point.
10. Choose confirm.


Information

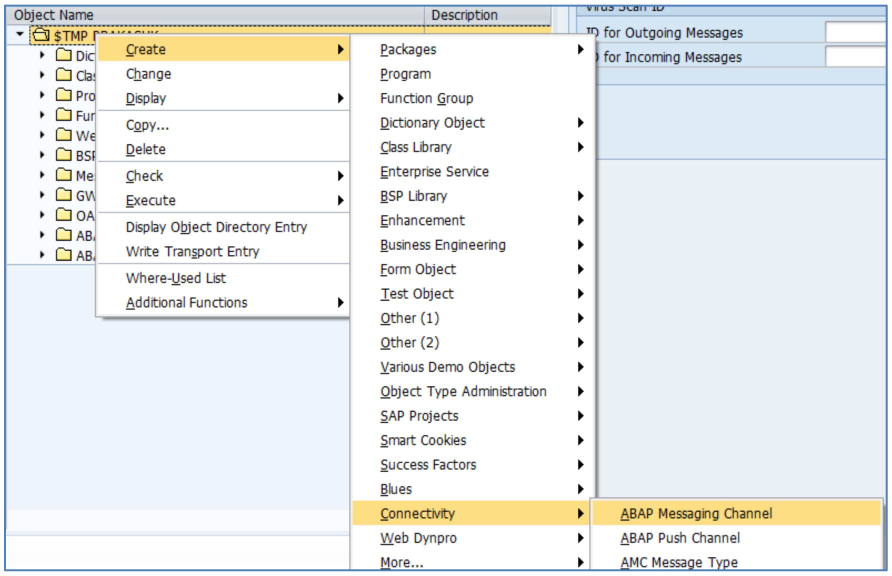
Generation of APC class CL_APC_WS_EXT_DEMO_OPEN_SALES_ and ICF service /sap/bc/apc/sap/demo_open_sales_apc was successful.

11. Open the generated service class ***CL_APC_WS_EXT_DEMO_OPEN_SALES*** and implement the methods (provide an empty implementation) of the interface ***IF_APC_WS_EXTENSION***.
12. Activate the class ***CL_APC_WS_EXT_DEMO_OPEN_SALES.***
13. Activate the APC Service.

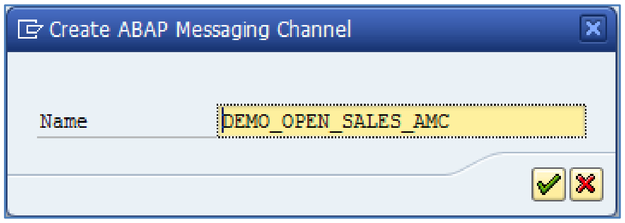
Upon activation of APC, a Service URL is generated and activated which will be used later on.

5.2.2 Creating ABAP Messaging Channel

1. Open ABAP Workbench (SE80).
2. Choose the required ABAP package.
3. In the context menu, choose ***Create*** >***Connectivity*** > ***ABAP Message Channel.***



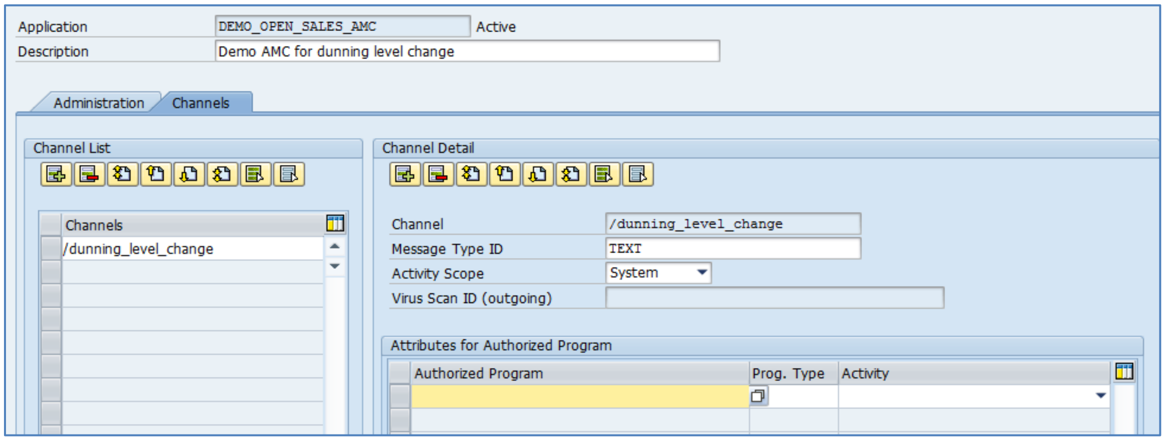
4. Enter the ABAP Message Channel name as “***DEMO_OPEN_SALES_AMC***”.


Create ABAP Messaging Channel

Name DEMO_OPEN_SALES_AMC

5. Provide the ***package*** and ***Transport Request*** number details (if required).

6. Enter the AMC ***description***.
7. On the Channel List panel, add a new channel with the name ***/dunning_level_change*** and press `Enter`.
8. On the Channel Detail panel, choose ***Message Type ID as TEXT*** (from F4 help available).
9. Choose ***Activity Scope*** as ***System***.


Application DEMO_OPEN_SALES_AMC Active
Description Demo AMC for dunning level change

/dunning_level_change

Channel /dunning_level_change
Message Type ID TEXT
Activity Scope System
Virus Scan ID (outgoing)

10. Save and Activate AMC.

5.3 Enable APC to Notify the Changes (dunning level change)

This section describes how the ABAP Message Channel is configured to notify the dunning level change from the ABAP server. For this, we need to create one simple report, which changes the dunning level and notifies the APC about the change.

5.3.1 Binding the ABAP Push Channel with ABAP Message Channel

1. Open ABAP Workbench (SE80).
2. In the APC service class, bind the AMC with APC.
3. Copy and paste the below code snippet to the method ***on_start***.

```
*********************************************************************************
* Code which bind AMC with APC
*********************************************************************************
DATA: lo_binding      TYPE REF TO if_apc_ws_binding_manager.
DATA: lx_error        TYPE REF TO cx_apc_error.
DATA: lv_message      TYPE string.
* bind the APC WebSocket connection to an AMC channel
    TRY.
        lo_binding = i_context->get_binding_manager( ).
        lo_binding->bind_amc_message_consumer(
                             i_application_id = 'DEMO_OPEN_SALES_AMC'
                             i_channel_id     = '/dunning_level_change' ).
      CATCH cx_apc_error INTO lx_error.
        lv_message = lx_error->get_text( ).
    ENDTRY.
*********************************************************************************
```

The code snippet provided above binds the ABAP Push Channel with ABAP Message Channel by providing the AMC and Channel ID to bind method.

5.3.2 Report to Modify the Dunning Level and Notify the Changes

1. Open ABAP Workbench (SE80).
2. Create an ABAP Report with the name ***RS_CHANGE_DUN_LEVEL***.
3. Copy and paste the below code snippet into the report.

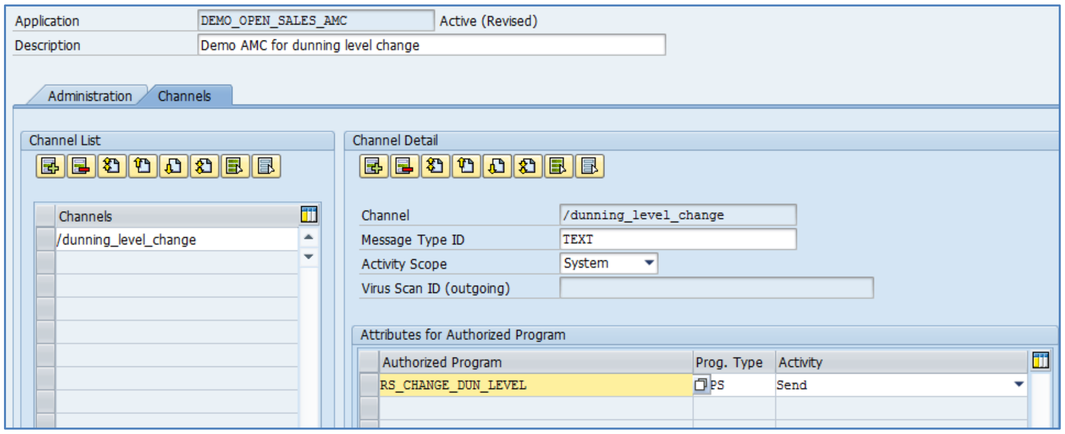
```
*********************************************************************************
* Code which changes the dunning level
*********************************************************************************

DATA(mo_dunning_engine) = NEW cl_oia_dunning_engine( ).
  mo_dunning_engine->get_dunning_list( RECEIVING rt_dunning_info =
DATA(lt_dunning_info) ).
  mo_dunning_engine->execute_dunning( it_dunning_info = lt_dunning_info ).


*********************************************************************************
* Code to invoke the APC to notify the dunning level change
*********************************************************************************
DATA:
      lo_producer       TYPE REF TO if_amc_message_producer_text,
      lv_message        TYPE string.

      lv_message = |Dunning level has been changed. Reload the page to view the
updated dunning levels|.

    TRY.
        lo_producer ?= cl_amc_channel_manager=>create_message_producer(
                     i_application_id = 'DEMO_OPEN_SALES_AMC'
                     i_channel_id     = '/dunning_level_change' ).
        lo_producer->send( i_message = lv_message ).
      CATCH cx_amc_error INTO DATA(lx_amc_error).
        RETURN.
    ENDTRY.
*********************************************************************************
WRITE : / 'Dunning Level Has been updated successfully'.
```

The code snippet provided above calls an API, which modifies the dunning level and then AMC is notified about the changes using the channel ID configured on the AMC.

5.3.3 Configure AMC to Listen and Notify the Dunning Level Change

1. Open ABAP Workbench (SE80).
2. Navigate to AMC ***DEMO_OPEN_SALES_AMC***.
3. Choose the Channel ID ***/dunning_level_change***.

4. In the section “***Attributes for Authorized Program***”, add the report that was created above for changing the dunning level.
5. Enter Authorized Program as ***RS_CHANGE_DUN_LEVEL.***
6. Enter Prog. Type as ***REPS.***
7. Choose Activity as ***R Send*** from the drop down.



Application	DEMO_OPEN_SALES_AMC	Active (Revised)
Description	Demo AMC for dunning level change	

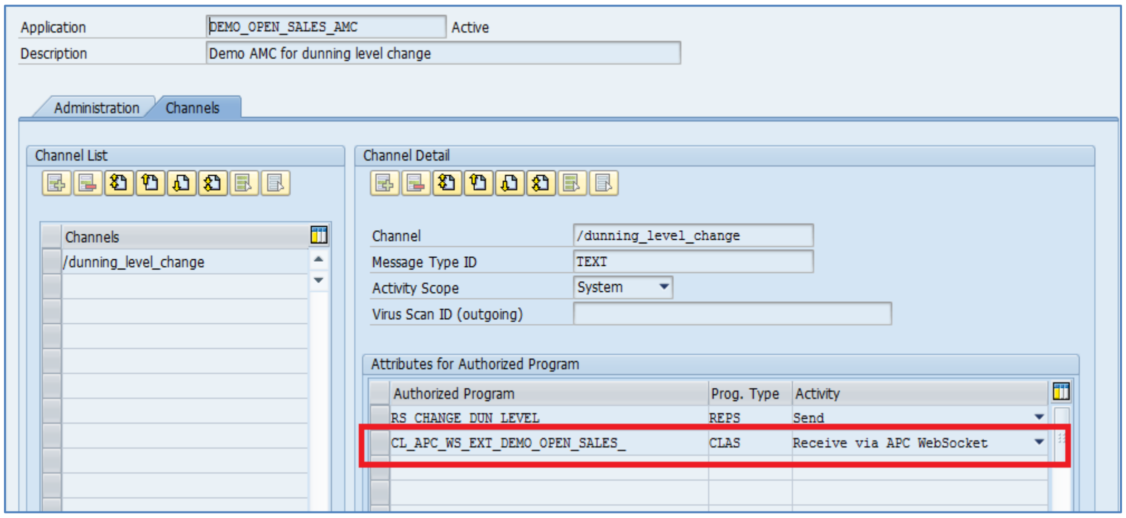
Channels
/dunning_level_change

Channel	/dunning_level_change
Message Type ID	TEXT
Activity Scope	System
Virus Scan ID (outgoing)	

Attributes for Authorized Program

Authorized Program	Prog. Type	Activity
RS_CHANGE_DUN_LEVEL	PS	Send

8. Enter Authorized Program as ***CL_APC_WS_EXT_DEMO_OPEN_SALES_***
9. Enter Prog. Type as ***CLAS***
10. Choose Activity as ***C Receive via APC WebSocket.***



Application	DEMO_OPEN_SALES_AMC	Active
Description	Demo AMC for dunning level change	

Channels
/dunning_level_change

Channel	/dunning_level_change
Message Type ID	TEXT
Activity Scope	System
Virus Scan ID (outgoing)	

Attributes for Authorized Program

Authorized Program	Prog. Type	Activity
RS_CHANGE_DUN_LEVEL	REPS	Send
CL_APC_WS_EXT_DEMO_OPEN_SALES_	CLAS	Receive via APC WebSocket

The APC and AMC are now configured to notify the changes when there is an update on dunning level.

The dunning level change from the report RS_CHANGE_DUN_LEVEL will only be notified by AMC as the AMC is configured accordingly. However, AMC will not notify if the dunning level is changed by any other sources.

5.4 Enable SAP UI5 Application to Receive the Notification from APC

This section describes how to enable the SAP UI5 application to listen to the notification from APC using Web Sockets.

5.4.1 Enable SAP UI5 Application to Listen to the Notification using Web Socket

To enable SAP UI5 application to receive/listen to the notification from APC, a Web Socket has to be implemented to listen to the APC service.

1. Open the ***OpenSalesOrder.controller.js*** file.
2. Create a new method called ***initDunningRunFeed***.
3. Copy and paste the below code snippet to the method.

```
initDunningRunFeed : function () {
    var hostLocation = window.location, socket, socketHostURI, webSocketURI;
    if (hostLocation.protocol === "https:") {
        socketHostURI = "wss:";
    } else {
        socketHostURI = "ws:";
    }
    socketHostURI += "//" + hostLocation.host;
    webSocketURI = socketHostURI + '/sap/bc/apc/sap/demo_open_sales_apc';
    try{
        socket = new WebSocket(webSocketURI);
        socket.onopen = function(){ };
        socket.onmessage = function(dunningRunFeed) {
            if (dunningRunFeed.data !== undefined ) {
                jQuery.sap.require("sap.m.MessageBox");
                sap.m.InstanceManager.closeAllDialogs();
                sap.m.MessageBox.show(
                    dunningRunFeed.data,
                    sap.m.MessageBox.Icon.INFORMATION,
                    "APC Notification",
                    [ sap.m.MessageBox.Action.OK ]
                );
            }
        };
        socket.onclose = function(){ };
    } catch(exception){ }
},
```

The code snippet provided above frames the URI to APC service and establishes a Web Socket to listen to the APC service notifications. The life cycle method ***onmessage*** displays a confirmation dialog when there is a notification, i.e. where there is a change in the dunning level.

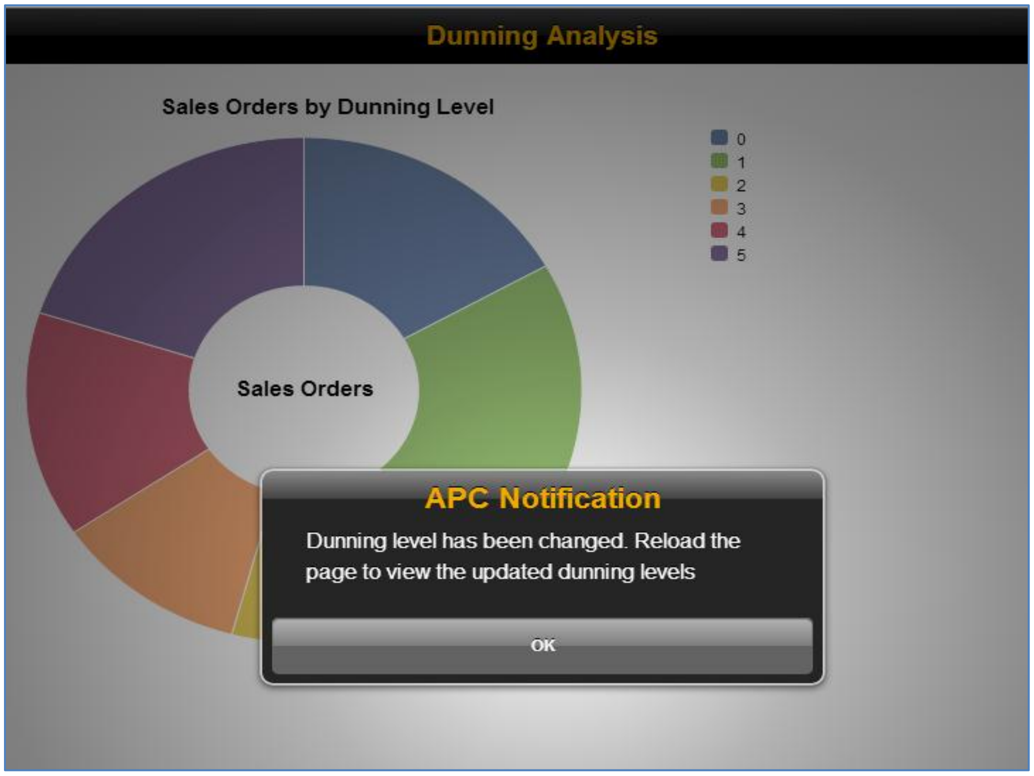
4. Call the ***initDunningRunFeed*** method from the ***onInit*** method of ***OpenSalesOrder.controller.js***.

```
onInit: function() {
    ………
    // call the method init the dunning level change feed
    this.initDunningRunFeed();
},
```

5.4.2 Test the SAP UI5 Application with APC Notification

To test the application end-to-end, the SAP UI5 application has to be launched and the report has to be executed. Upon executing the report, the SAP UI5 application displays a dialog that displays the message delivered from APC.

1. Launch the SAP UI5 application (on chrome with simulator iPhone or iPad).
2. Run the ABAP report program ***RS_CHANGE_DUN_LEVEL***.
3. A dialog appears on the SAP UI5 application to display the message from APC.



Dunning level has been changed. Reload the page to view the updated dunning levels

6. Appendix

6.1 Appendix A – Abbreviations

Abbreviation	Description
EPM	Enterprise Procurement Model
NW	NetWeaver
ABAP AS	ABAP Application Server
APC	ABAP Push Channel
AMC	ABAP Messaging Channel
GW	Gateway
ICF	Internet Communication Framework
URI	Uniform Resource Identifier

6.2 Appendix B – References

- The NetWeaver Enterprise Procurement Model - An Introduction to EPM: http://scn.sap.com/docs/DOC-31458

6.3 Appendix C – Link to ABAP for SAP HANA Reference Scenarios

- ABAP for SAP HANA Reference Scenario - http://scn.sap.com/docs/DOC-35518
- ABAP for SAP HANA Reference Scenario: Implementing Open Items Analysis using SAP Floor Plan Manager - http://scn.sap.com/docs/DOC-47390
- ABAP for SAP HANA Reference Scenario: Implementing Mobile based Open Items Analysis Using SAP UI5 and SAP Gateway Foundation Component - http://scn.sap.com/docs/DOC-47388

6.4 Appendix D – Links to SAP NetWeaver Gateway Documents

- SCN Link: http://scn.sap.com/community/netweaver-gateway
- How-To Guides: http://scn.sap.com/docs/DOC-27405

www.sap.com