

# BRING THE FULL SAP FIORI USER EXPERIENCE TO YOUR SCREENS IN SAP GUI

In this session, we will transform plant maintenance transactions IW29 and IW53, making them appear like SAP Fiori applications. The goal is to provide the look, feel and main functionality of such an application, down to the responsive features and formatting which provides a much more delightful and modern experience than the original SAP GUI transactions. Simplifications to screens will tailor the backend transactions to the needs of our specific user group.

Since our target audience mostly use mobile devices, we will focus on tablet users first and render screens via Slipstream Engine, built into SAP Screen Personas. Then, we will extend our scenario to cover phone users as well.

Use the following URL to access the training system:

<http://link.personas.help/opensap/se>

You will be assigned a two-digit user number, between 01 and 99. Log on with the following credentials:

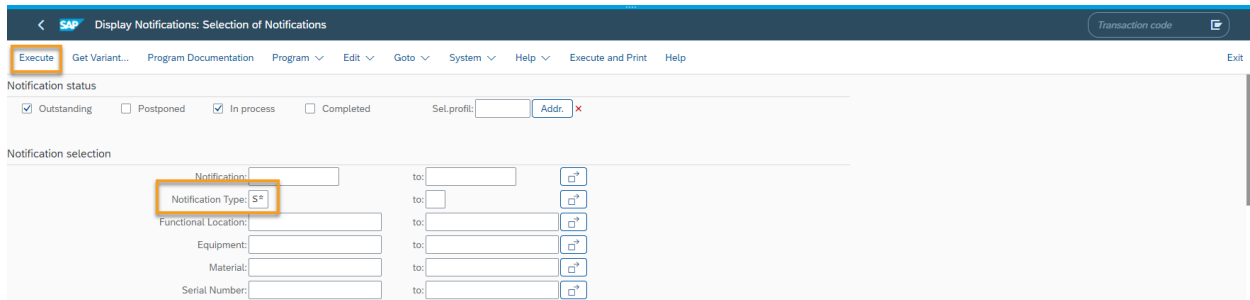
User:               TECHED-xx (where xx is your user number)  
Password:       welcome

In the exercises, purple frames show the objects that should be (multi)selected. Amber frames or simple highlighting point to objects of interest or functions you should click on after such selections.

If you have a chance, we recommend using two screens. One should display this workbook you downloaded locally, so you can easily zoom in or out as necessary, to see all details. The other screen is to perform the actual hands-on tasks. This arrangement significantly improves your experience, instead of having to switch between windows on a single, small screen.

## First task: Improving the look and usability of the Notification List

Let's start with IW29 which allows selection of notifications. Our users are interested in service notifications, so enter S\* as the Notification Type selection, and hit 'Execute' to get the result list:

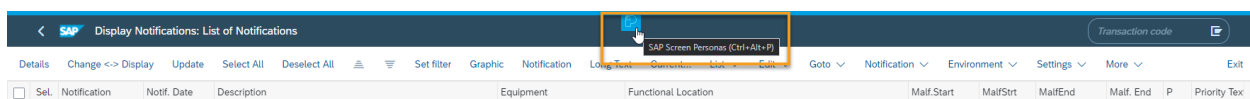


The screenshot shows two SAP screens. The top screen, 'Display Notifications: Selection of Notifications', has a menu bar with 'Execute' highlighted. Below it, 'Notification status' shows 'In process' selected. The 'Notification selection' section has 'Notification Type: S\*' entered. The bottom screen, 'Display Notifications: List of Notifications', shows a table of notifications with columns for selection, notification ID, date, description, equipment, functional location, and start/end times. A 'Cancel' button is at the bottom right.

Sel.	Notification	Notif. Date	Description	Equipment	Functional Location	Mailf.Start	MailfStrt	MailfEnd	Mailf. End	P	Priority Text
<input type="checkbox"/>	300003236	07/26/2020	TESTING THE NOTIFICATION DESCRIPTION	10000002		07/26/2020	05:33:42		00:00:00	1	1-Very High
<input type="checkbox"/>	300003409	09/07/2020		10000002		09/07/2020	05:57:43		00:00:00	3	3-Medium
<input type="checkbox"/>	300003412	09/07/2020		10000002		09/07/2020	06:06:02		00:00:00	4	4-Low
<input type="checkbox"/>	300003413	09/07/2020		10000002		09/07/2020	06:07:28		00:00:00	2	2-High
<input type="checkbox"/>	300003414	09/07/2020		10000003		09/07/2020	06:12:43		00:00:00		
<input type="checkbox"/>	300003416	09/07/2020		10000003		09/07/2020	12:43:06		00:00:00		
<input type="checkbox"/>	300003418	09/07/2020		10000003		09/07/2020	13:15:59		00:00:00		
<input type="checkbox"/>	300003425	09/07/2020				09/07/2020	16:07:06		00:00:00		
<input type="checkbox"/>	300003432	09/09/2020	Asistencia técnica	10000002		09/09/2020	14:20:07		00:00:00		
<input type="checkbox"/>	300003433	09/09/2020	Servicio 01	10000002		09/09/2020	14:55:36		00:00:00		
<input type="checkbox"/>	300003435	09/09/2020	Asistencia técnica	10000002		09/09/2020	18:26:48		00:00:00		
<input type="checkbox"/>	300003451	09/10/2020	Test de personas	10000002		09/10/2020	12:40:20		00:00:00		
<input type="checkbox"/>	300003461	09/14/2020	Avería en el servicio	10000002		09/14/2020	16:29:03		00:00:00		
<input type="checkbox"/>	300003463	09/14/2020	Test de avería técnica	10000002		09/14/2020	16:34:34		00:00:00		
<input type="checkbox"/>	300003504	09/24/2020	Test	210100024	1010-SPA-SAC-PLAR2-CMT1	09/24/2020	23:30:21		00:00:00		
<input type="checkbox"/>	300003505	09/24/2020	Brake failure	210100024	1010-SPA-SAC-PLAR2-CMT1	09/24/2020	23:35:11		00:00:00	3	3-Medium
<input type="checkbox"/>	300003563	10/03/2020				10/03/2020	15:40:50		00:00:00		
<input type="checkbox"/>	300003594	10/09/2020				10/09/2020	20:26:22		00:00:00		

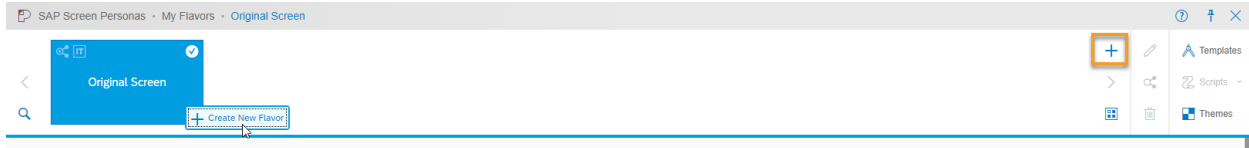
This is presented as a grid with many columns, from which a lot are not relevant for our target user group. We could change this list with regular flavor editing means, but our goal is to provide the result in a format resembling an SAP Fiori application. In other words, we want to transform it into a SAPUI5 table.

To do this, let's create a flavor first. Open the Flavor Manager by hovering over the top blue line and clicking on the Personas "P" in the middle, or by clicking on the "P" in the lower left corner.



The screenshot shows the 'List of Notifications' screen. A blue bar at the top contains icons for 'Details', 'Change <-> Display', 'Update', 'Select All', 'Deselect All', 'Set filter', 'Graphic', 'Notification', 'Long Text', 'Current...', 'List', 'Edit', 'Goto', 'Notification', 'Environment', 'Settings', 'More', and 'Exit'. A yellow box highlights the 'Notification' icon, which has a tooltip that reads 'SAP Screen Response (Ctrl+Alt+P)'.

Right-click on the 'Original Screen' tile and select 'Create New Flavor'. Alternatively, you can also click the plus sign on the right side:



Specify a flavor name and description (can be anything, doesn't have to be unique), then hit 'Create':

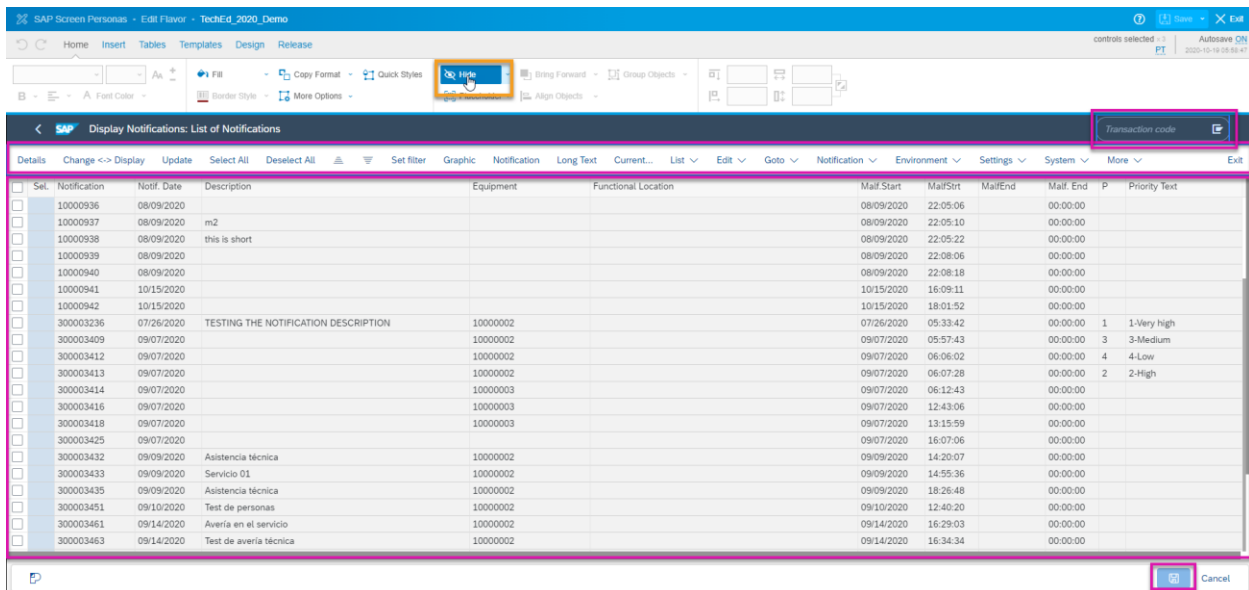
**New Flavor** ✕

Create a new flavor from scratch.

Name \*

Description \*

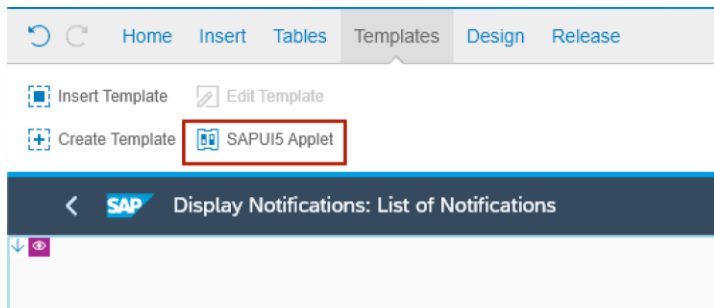
First, let's remove the controls that we will replace or won't need. Multi-select (hold down Ctrl, then left click the objects, OR hold down Shift, then draw a box around the objects by holding the left mouse button, then release) the following: the 'Transaction code' command field, menu bar, the whole grid and the Save button in the footer. Then click 'Hide' in the Home tab:



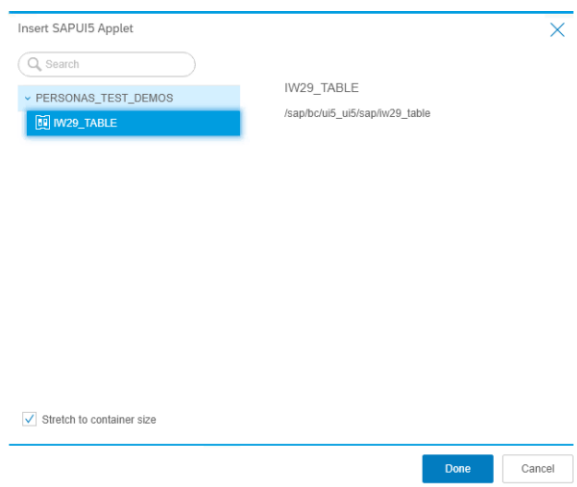
To render the table data using a SAPUI5 table, we will take advantage of a small SAPUI5 application (applet) deployed in the backend system. Such an applet is developed using appropriate tools such as SAP WebIDE, SAP Business Application Studio or Visual Studio Code etc.

In our case, the applet is already deployed so we only need to reference it in our flavor. Applets are stored in the SAPUI5 ABAP Repository as BSP applications.

Select the 'Templates' tab, then 'SAPUI5 Applet':

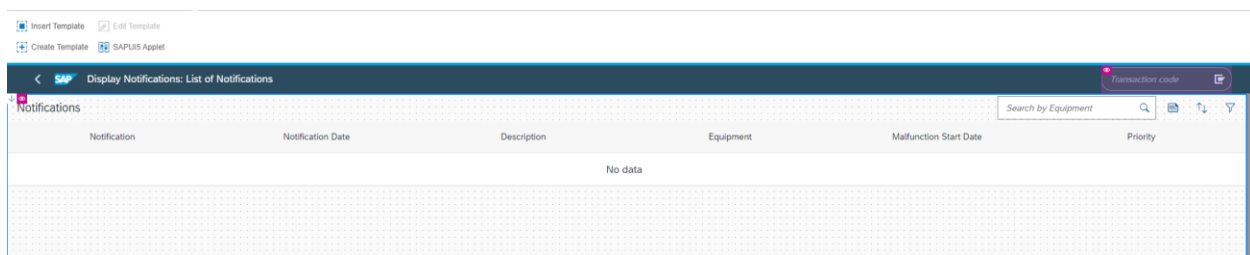


Look for the package containing the SAPUI5 applet specified during the deployment. In our case, this is PERSONAS\_TEST\_DEMOS. Select the desired SAPUI5 applet component:

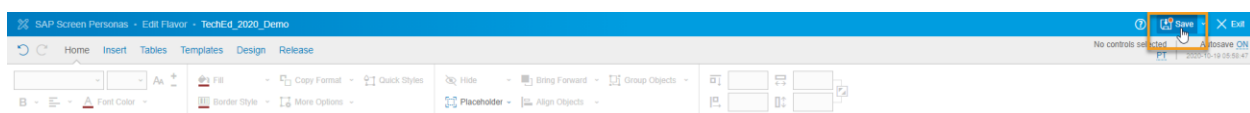


Check the 'Stretch to container size' option. Doing so will enable the SAPUI5 applet view to take all the available container size and ensure that if the container has been resized, the SAPUI5 applet will dynamically adapt its size based on the screen resolution of the device.

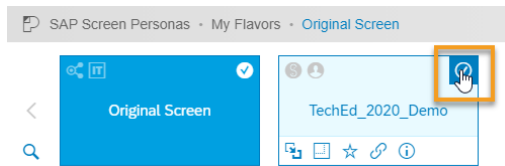
Press 'Done' to insert the SAPUI5 applet into the *GuiUserArea* container. At this point, you will see the SAPUI5 applet appear as part of your flavor:



Hit 'Save', then exit the editor.



Set this new flavor as your default by opening the Flavor Manager and clicking on the check mark in the upper right corner of the flavor tile:

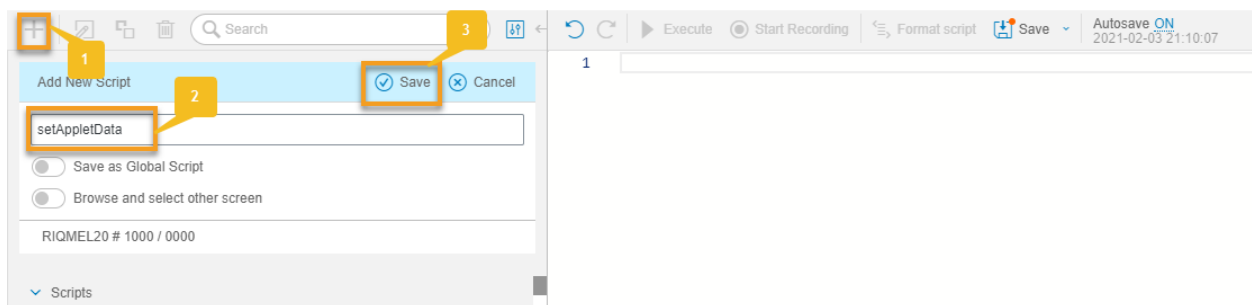


The SAPUI5 applet component will now be rendered in IW29 runtime, however the notifications list will be empty. To bind the data between the SAP GUI runtime and the SAPUI5 applet, we can use the SAP Screen Personas Scripting Engine API.

Open the script editor:



Create a new script called **setAppletData**:



Insert the following source code:

```
// Load table data from screen
var fnReadModelData = function () {
    var oSelectedTable = session.findById("wnd[0]/usr/cnt1GRID1/shellcont/shell");
    var aColumns = oSelectedTable.columns;
    var aContents = [];
    var sSAPColumnName;
    var iTopRow, iRowIndex, oRow, i;
    if (oSelectedTable.rowCount > 0) {
        oSelectedTable.firstVisibleRow = 0;
        iTopRow = oSelectedTable.visibleRowCount - 1;
        for (iRowIndex = 0; iRowIndex < oSelectedTable.rowCount; iRowIndex++) {
            oRow = {};
            if (iRowIndex > iTopRow) {
                if (iTopRow + oSelectedTable.visibleRowCount > oSelectedTable.rowCount) {
                    oSelectedTable.firstVisibleRow = oSelectedTable.rowCount - oSelectedTable.visibleRowCount;
                } else {
```

```

        oSelectedTable.firstVisibleRow = iTopRow + 1;
    }
    iTopRow += oSelectedTable.visibleRowCount;
}
for (i = 0; i < aColumns.length; i++) {
    sSAPColumnName = aColumns.elementAt(i).name;
    oRow[sSAPColumnName] = oSelectedTable.getCellValue(iRowIdx, sSAPColumnName);
}
aContents.push(oRow);
}
}
return aContents;
};

var oComponent = session.findById(<PLACE YOUR APPLET ID HERE>).getComponent();
oComponent.getService("SelectionService").attachSelect(function(oEvent){
    session.utils.put("SELECTED_NOTIFICATION", oEvent.getParameter("id"));
    //Call the script that handles the SAPUI5 Fiori table selection event
    session.utils.executeScriptAsync("wnd[0]/scprtPersonas_0050568405451EDABE9DABE7B06B8923");
}).bind(this));

//Set the SAPUI5 Applet view model data
oViewModel = oComponent.getModel("DATA");
oViewModel.setData({"NotificationCollection" : fnReadModelData()});

//Restore default selected row on navigation
var sSelectedNotification = session.utils.get("SELECTED_NOTIFICATION");
if(sSelectedNotification){
    oComponent.getService("SelectionService").setSelectedRowById(sSelectedNotification);
    session.utils.put("SELECTED_NOTIFICATION", "");
}

```

Use the Object Selector in the right pane and select the SAPUI5 applet control to find its control ID, then replace the `<PLACE YOUR APPLET ID HERE>` placeholder in the code snippet above.


Example of how that code snippet should look like:

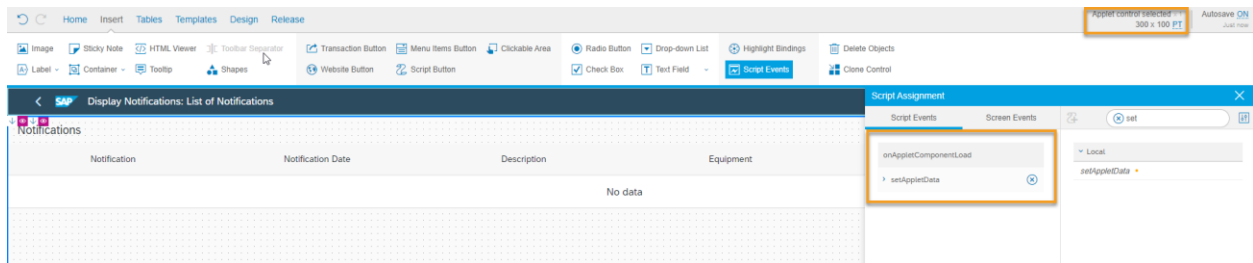
```

var oComponent = session.findById("wnd[0]/usr/appPersonas_161121438708911").getComponent();

```

Now we need to assign the created script to the *onAppletComponentLoad* event.

Save the script and leave the script editor by clicking  in its upper right corner and go back to the flavor editor. Select the applet control. In the 'Insert' tab, click on 'Script Events', then for the *onAppletComponentLoad* event, start typing *set...* and select the **setAppletData** script by clicking on the arrow on the right or dragging its name to the event:



The data binding between the SAPUI5 applet and the IW29 runtime is now complete. Save the flavor and exit the editor. Don't be alarmed by the blank screen, this is expected. Return to the selection screen by clicking the 'Back' button in the upper left corner, then re-run the report by hitting 'Execute'. Now the script will run, and the result is presented as a beautiful SAPUI5 table, complete with sorting, filtering and search capabilities.

SAP Display Notifications: List of Notifications						
Notifications						
Notification	Notification Date	Description	Equipment	Malfunction Start Date	Priority	
300003236 S3	07/26/2020	TESTING THE NOTIFICATION DESCRIPTION	10000002	07/26/2020	1-Very High	
300003409 S3	09/07/2020		10000002	09/07/2020	3-Medium	
300003412 S3	09/07/2020		10000002	09/07/2020	4-Low	
300003413 S3	09/07/2020		10000002	09/07/2020	2-High	
300003414 S3	09/07/2020		10000003	09/07/2020		
300003416 S3	09/07/2020		10000003	09/07/2020		
300003418 S3	09/07/2020		10000003	09/07/2020		
300003425 S3	09/07/2020			09/07/2020		
300003432 S3	09/09/2020	Asistencia técnica	10000002	09/09/2020		

The SAPUI5 applet is also coded to respond to line selection. When clicking on a row in the table, it will run transaction IW53, displaying the selected notification.

Of course, that screen looks like a regular SAP GUI transaction rendered by Slipstream Engine, so we've got to do something about the look and feel of it to match our notification list. This is what we start with:

**SAP** Display Service Notification: Service Request Transaction code

Display object Partners Address... Document flow Action log Default values Status Organization... Services for Object Service notification Edit Goto More Exit

Notification: 300003236 S3 TESTING THE NOTIFICATION DESCRIPTION

Notific. Status: OSNO

**Notification** Additional data 1 System availability Malfunction, breakdown Location data

**Cust. address** Contact person address Message address Obj. address

Sold-To Party: USCUL05 Bluestar Corp

Street/Hse No.: 25 Waverley St.

Location: 29401 CHARLESTON US SC

Telephone: Fax:

PartnerTimeZone: 10/22/2020 04:00:59 EST

**Additional Data**

Reported By: Date: 07/26/2020 05:33:41

Cust. Reference: Sales Doc.: 0

**Reference Object**

Functional loc.: Equipment: 100000002 Trad.Good 20, Reorder Point Assembly:

**Subject**

Cancel

There are plenty of opportunities for improvement, so let's do that next.

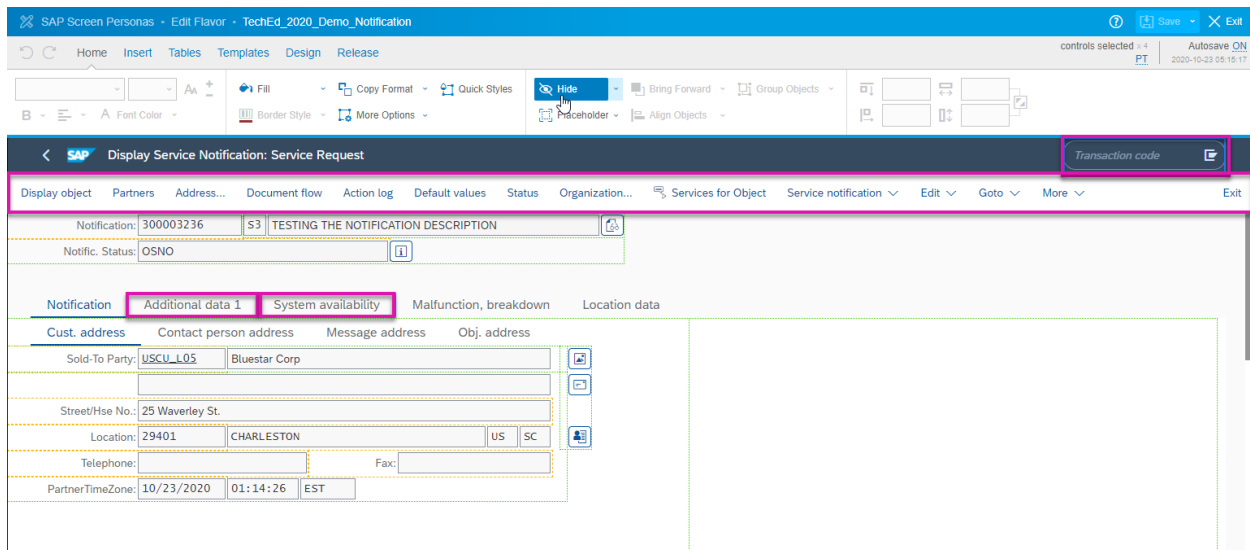


## Second task: Turn the Display Notification screen into an SAP Fiori object page

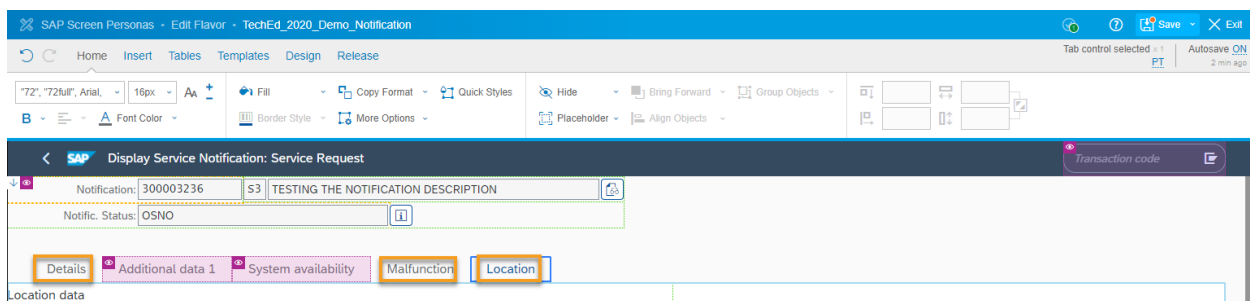
The Display Notification screen shown above is rather busy, and we want to provide our users a simpler, better looking application that adheres to the SAP Fiori design guidelines.

During our Design Thinking sessions with our users, we concluded that they are interested in information which can be grouped into general details, the malfunction data and the location of the equipment the notification is about.

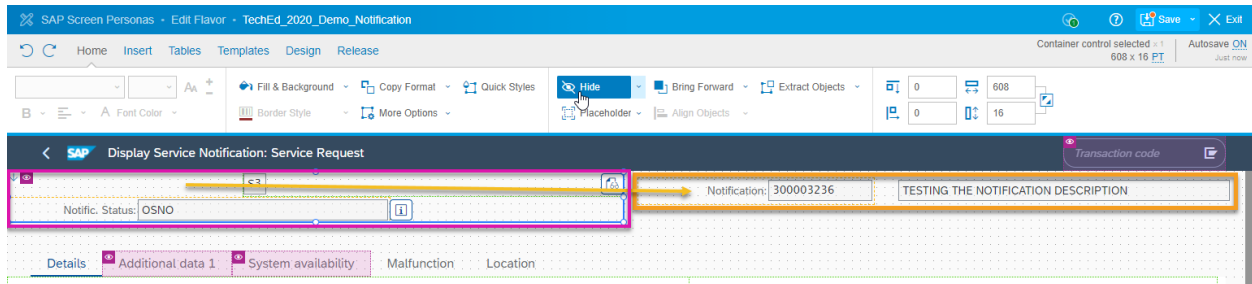
Create a new flavor for this transaction. You did this earlier with IW29, so follow the same steps. As the first change, let's hide the 'Transaction code' field and the menu bar. We can also hide the 'Additional data 1' and 'System availability' tabs, because our users don't need them.



Let's rename the three remaining tabs. Double-clicking its header switches to a tab, and by repeated double-click we can change the label to Details, Malfunction and Location, respectively.

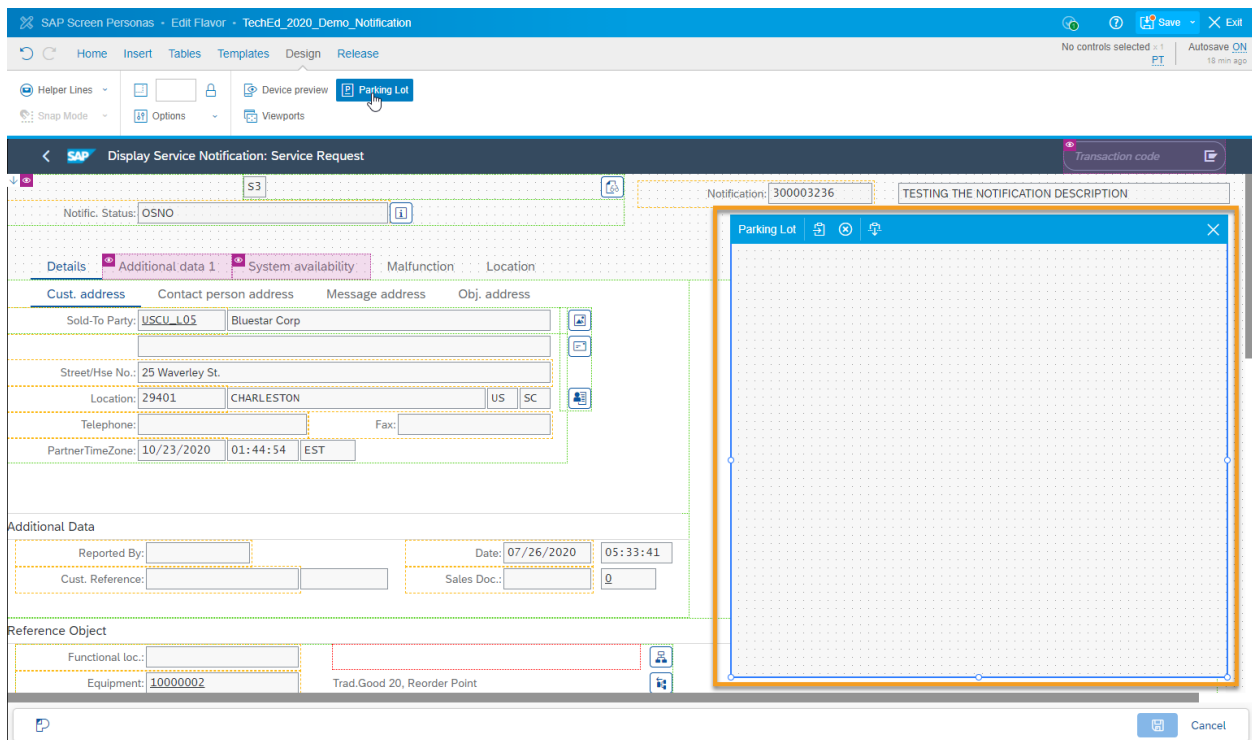


In the header area, we need the notification number with its label and the short description, everything else should be hidden. Let's grab these three controls and drag them aside to the right, where there is some free space. Then, we can hide the container for the other header fields:



Switch back to the 'Details' tab. From here, we will need a few objects. When simplifying tabs, we can once again drag them to an empty area outside the tab strip before moving them to their correct place. However, in our case this is a little more difficult because the tab strip is wide and our screen may not be big enough, resulting in a lot of horizontal scrolling. In this scenario, a great tool to collect these objects in an intermediary location is the so-called 'Parking Lot' which is something like a clipboard where we can temporarily store controls, then place them where we want them later.

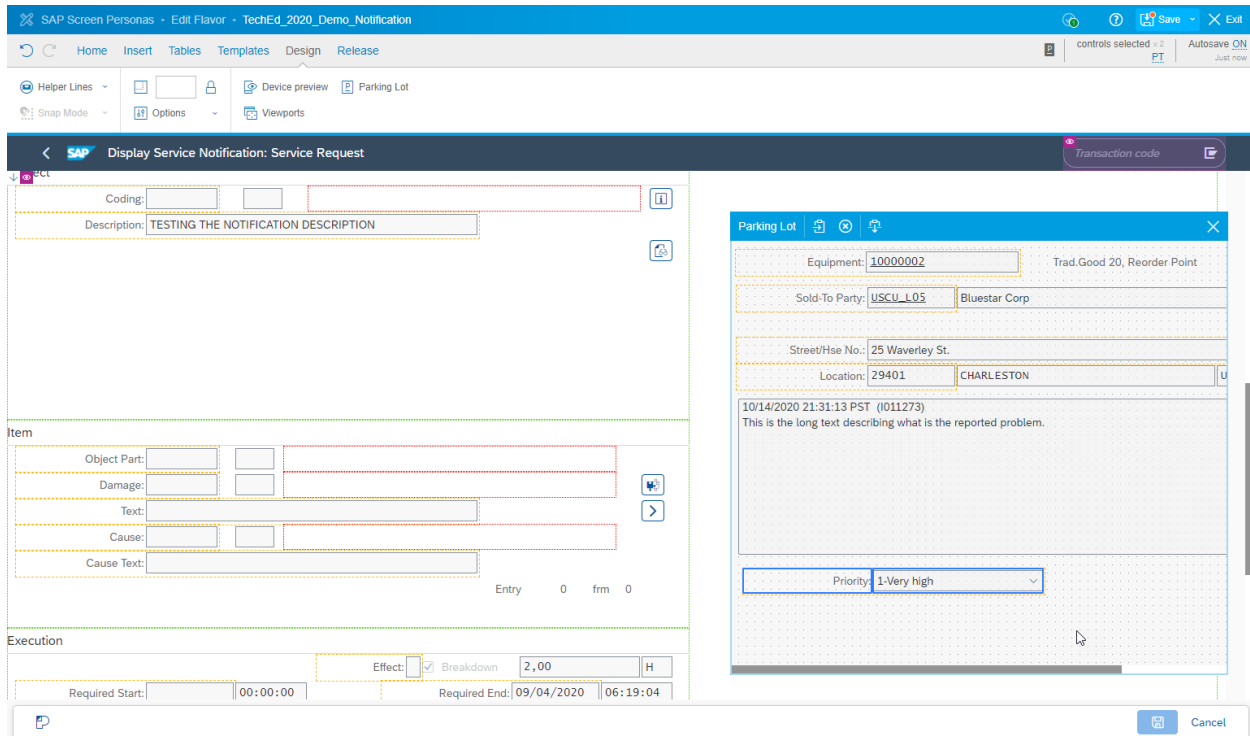
In the Flavor manager, go to tab 'Design', then click 'Parking Lot'. An additional window opens, which we can move or resize as necessary, and move objects to it from the screen:



Let's move the following to the Parking Lot:

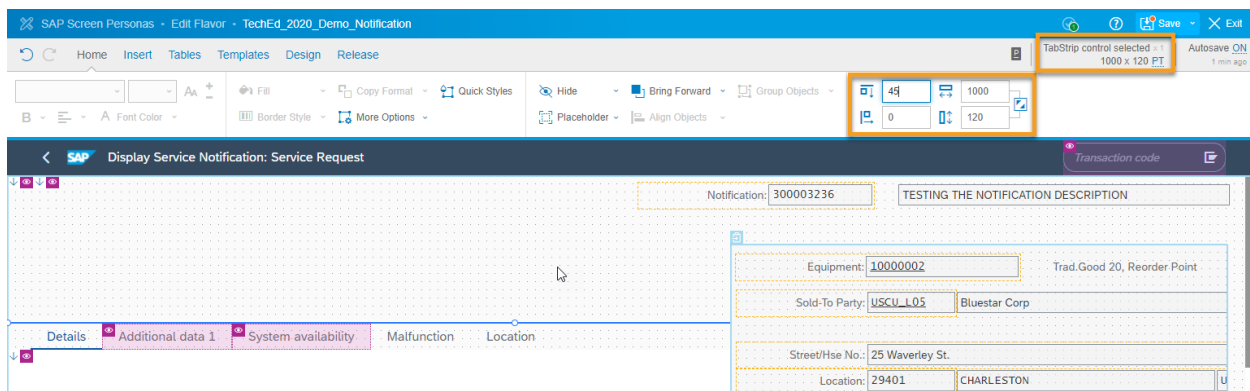
- The Sold-To Party label, ID, name
- Street/House no. label and field

- Location label, ZIP code, city, country, state
- From the 'Reference Object' group box:
  - o Functional loc. label, ID and description
  - o Equipment label, number and description
- From the 'Subject' group box, the long text
- From the 'Execution' group box, the Priority label and dropdown



Now, let's go ahead and remove all remaining controls in this tab so we end up with an empty 'Details' tab. Simply (multi)select those container controls / group boxes and hide them, along with their content.

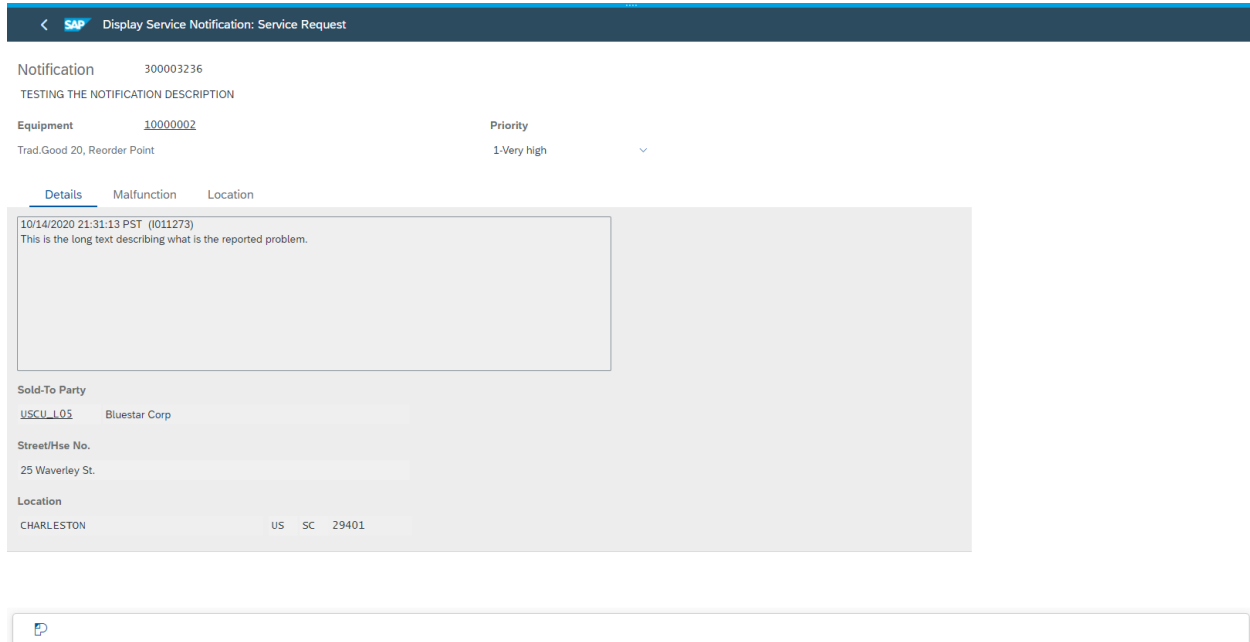
As the next step, resize and move down the complete tab strip control to make some more space for the header. Ensure that you select the correct object. The recommended values are:



It's time to move the 'parked' controls to their proper location. The equipment number and priority go to the header area above the tab strip. The others should be moved to the 'Details' tab, except for the

Functional Location, which we will use later. Don't forget, you can drag and resize the Parking Lot as necessary, if it's in the way or if something in it is not visible.

Relocate the header fields and format everything so the screen layout looks like the following screen shot.



Notification 300003236  
TESTING THE NOTIFICATION DESCRIPTION

Equipment 10000002 Priority 1-Very high

Trad.Good 20, Reorder Point

Details Malfunction Location

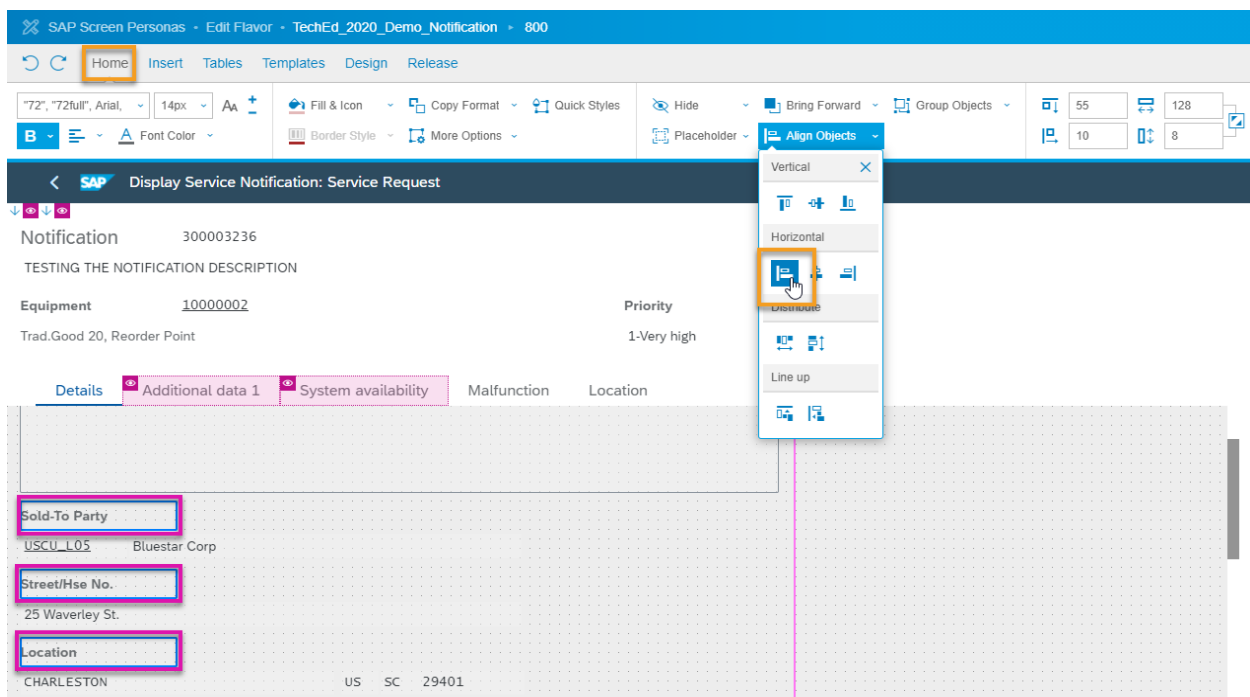
10/14/2020 21:31:13 PST (001273)  
This is the long text describing what is the reported problem.

Sold-To Party  
USCU\_L05 Bluestar Corp

Street/Hse No.  
25 Waverley St.

Location  
CHARLESTON US SC 29401

Field labels should be bold (use multi-select here to change the font decoration). You can take advantage of the various alignment options to ensure that labels and output fields are positioned properly. For example, to align the field labels, you can select them and set them to be left-aligned:



SAP Screen Personas - Edit Flavor - TechEd\_2020\_Demo\_Notification - 800

Home Insert Tables Templates Design Release

Font Size: 14px Font Color: [Color Picker]

Fill & Icon Copy Format Quick Styles Hide Bring Forward Group Objects Placeholder

Align Objects

Vertical Horizontal Line up

Display Service Notification: Service Request

Notification 300003236  
TESTING THE NOTIFICATION DESCRIPTION

Equipment 10000002 Priority 1-Very high

Trad.Good 20, Reorder Point

Details Additional data 1 System availability Malfunction Location

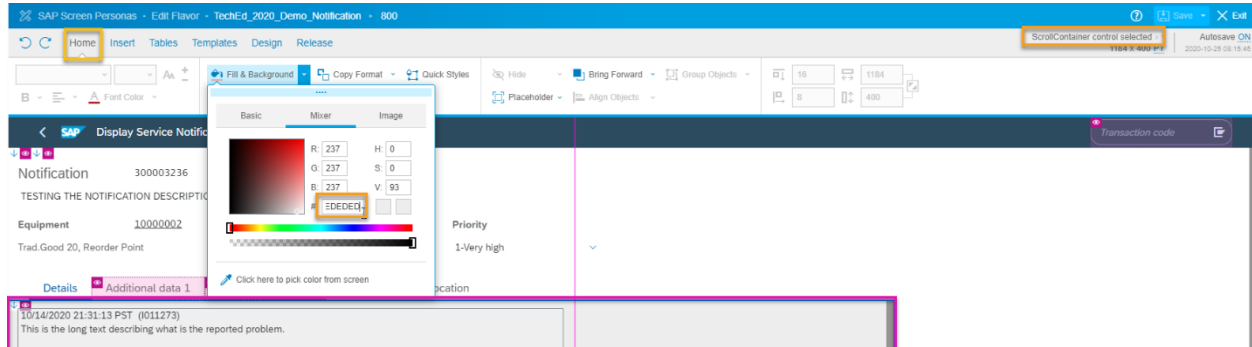
Sold-To Party  
USCU\_L05 Bluestar Corp

Street/Hse No.  
25 Waverley St.

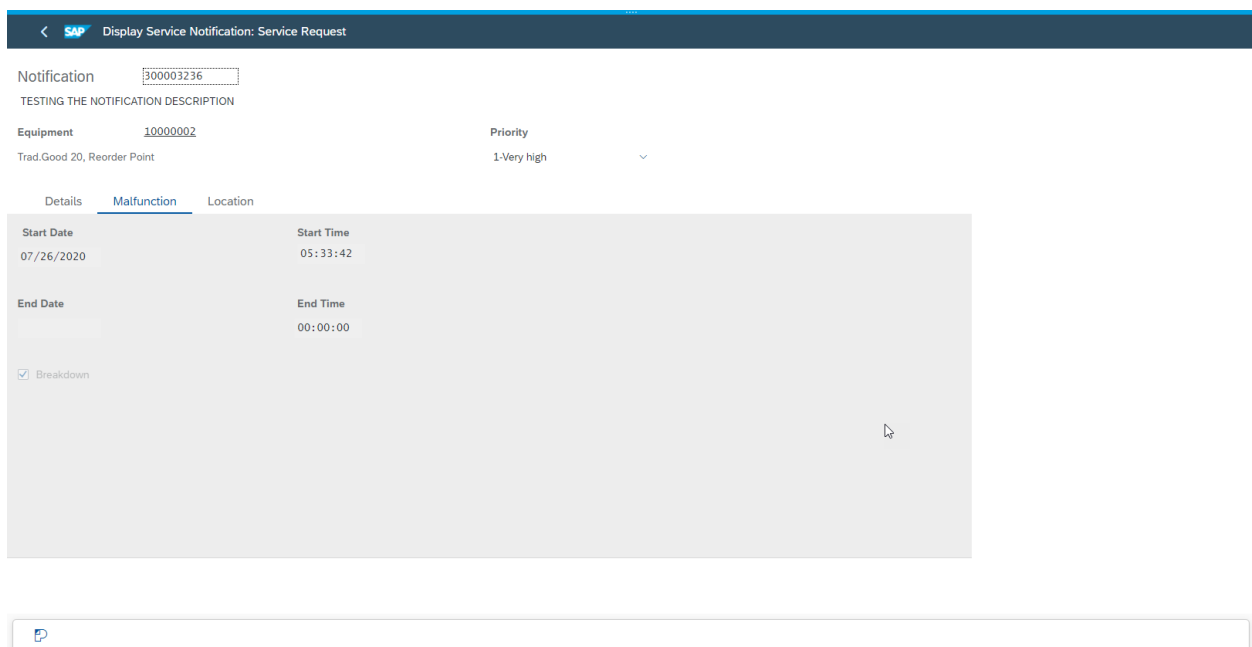
Location  
CHARLESTON US SC 29401

You should also remove the borders for input/output fields, which is possible using the 'Border Style' function in the 'Home' tab.

To get the gray tab background color, select the *ScrollContainer* control in the tab, then 'Fill and Background' in the Personas 'Home' tab. Go to 'Mixer' and enter the hex code EDEDED in the # field.



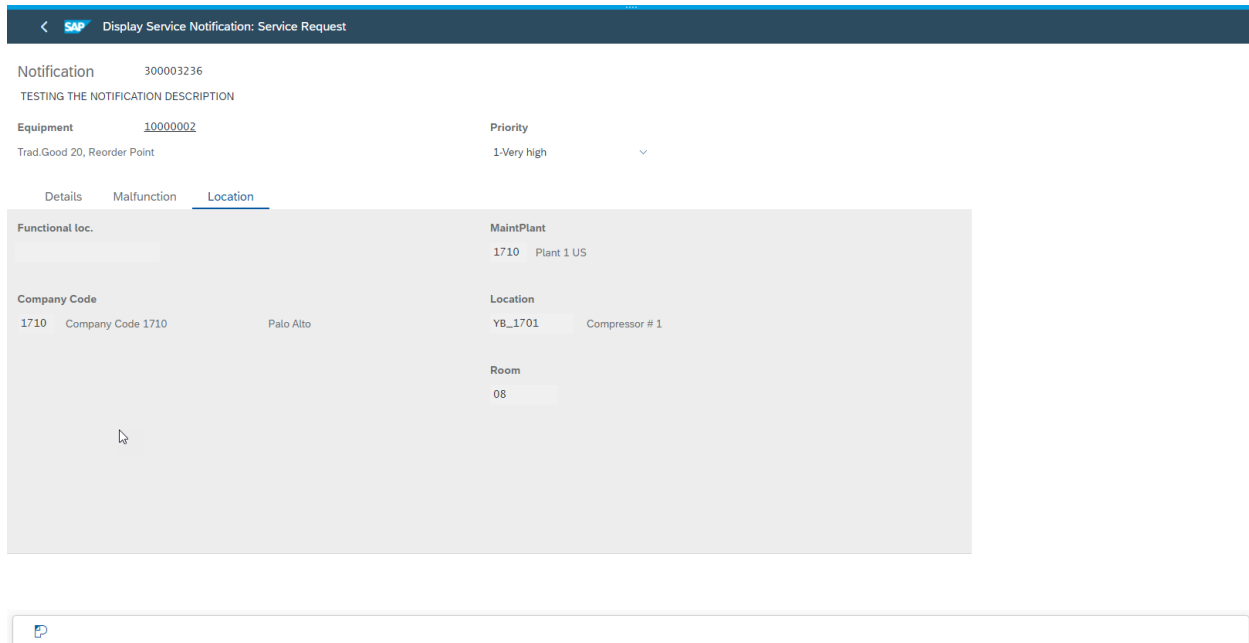
With this, the first tab is done. Let's move to the 'Malfunction' tab. This is simpler; all we need is reorganize and reformat the controls like this:



Steps:

- Change field labels (by double-clicking)
- Move the necessary controls out of their group box containers
- Hide the group boxes
- Arrange the objects according to the screen shot above
- Set the tab background color like earlier


Last editing step is the 'Location' tab. Here, the process is like with the 'Malfunction' tab, however we still have the 'Functional Location' field and description sitting in the Parking Lot. This is when they should be moved to the 'Location' tab, so the final look of this tab is as follows:

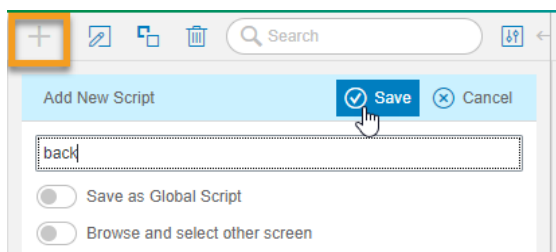


With this, the layout editing part of our flavor is complete. We significantly reduced complexity of this screen and gave it the look of an SAP Fiori application.

There is one problem remaining though. We would certainly want to be able to return to our notification list and select another document for display. With the original form of IW29 going to IW53, this is provided automatically. However, since we replaced the list grid in IW29 with the SAPUI5 table, this is not working anymore. When we selected a document, a script called IW53 as a new transaction, passed on the selected notification number and skipped the first screen. This means that if we now click the 'Back' button in IW53, we end up on the selection screen of that transaction, instead of returning to the IW29 list. We need to fix this.

Essentially, instead of the normal 'Back' button functionality, we need to re-run transaction IW29 and get past its selection screen. This is simple to do with a script, which we will attach to the standard 'Back' button.

Save and exit the flavor editor, then open the script editor. Create a new script by pressing the  sign and entering a name.



You could use recording to capture the steps, or here is the script to copy/paste:

```
// Start IW29
session.findById("wnd[0]/tbar[0]/okcd").text = "/nIW29";
session.findById("wnd[0]").sendVKey(0);

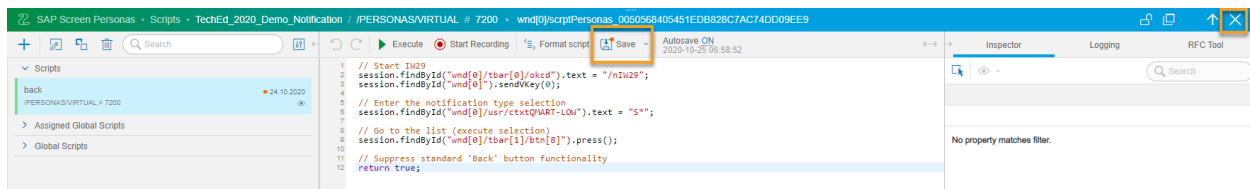
// Enter the notification type selection
session.findById("wnd[0]/usr/ctxtQMART-LOW").text = "S*";

// Go to the list (execute selection by pressing F8)
session.findById("wnd[0]/tbar[1]/btn[8]").press();


// Suppress standard 'Back' button functionality
return true;
```

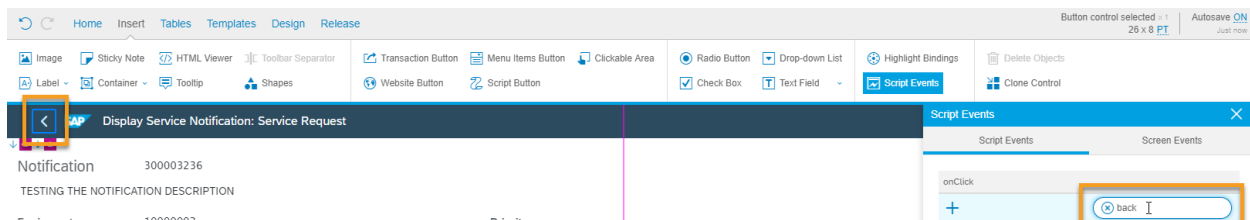
The last step is necessary, since we want to override the standard 'Back' button functionality with our script, and this will do exactly that.

Save the script, exit the script editor:



Go back to the flavor editor. Now we need to link this script to the 'Back' button.

Select the  'Back' button in the upper left corner, switch to the 'Insert' tab in the Flavor Manager and click 'Script Events'. Attach the new 'back' script to the *onClick* event.



Save and exit the Editor. Once again, set your new flavor as your default for this transaction by clicking the check mark in the upper right corner of the flavor tile. This is important to ensure that navigation between the Notification List and the individual Notification Display seamlessly uses your newly created flavors.

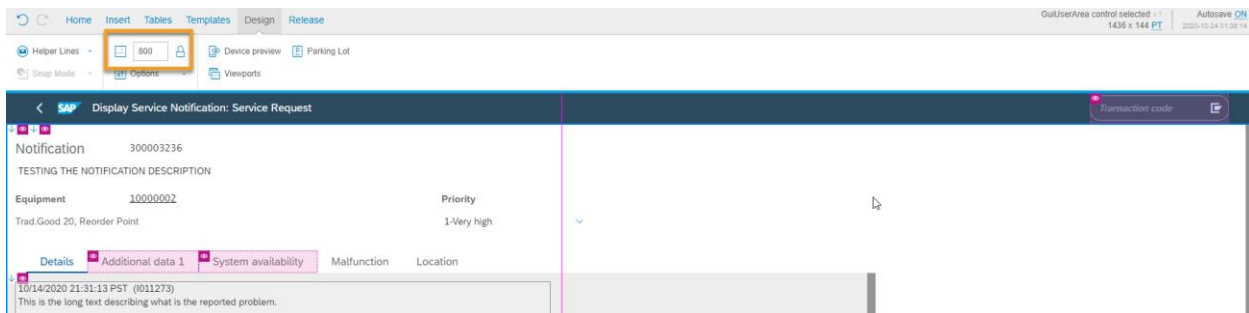
Now, if you click the 'Back' button, it will take you back to the IW29 list. Since the SAPUI5 app's context was saved, it knows exactly where you were when selecting the notification, so it can restore the list to that state.

### Third task: “Display Notification” flavor for phone users

Now that we handled users with tablets (or a desktop), let’s cater to those who are running this process on a phone. The screen is smaller, and its typical orientation is portrait rather than landscape. The new IW29 look with the SAPUI5 table is automatically responsive, rearranging the layout as necessary. This is not happening with our IW53 flavor, so we need to take care of this with a so-called adaptive flavor, designed for the phone screen. These can be considered as children of a flavor, which are automatically picked based on the actual screen width. We can have as many such adaptive flavors as necessary, for different screen sizes / devices. Our targeted phone model will be an iPhone 8.

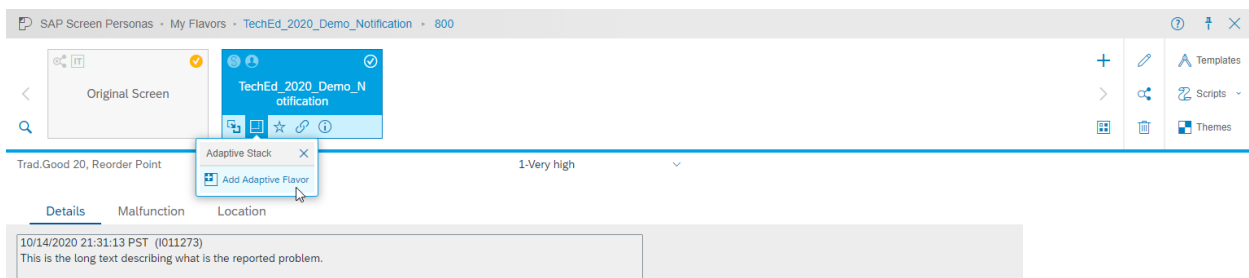
For this to work, it is essential that each flavor in the adaptive stack has its minimum width set, including the parent flavor. So, let’s change our tablet (parent) flavor and set its width to 800. This means that this flavor will be selected if the screen size is at least this wide.

Open the flavor editor and go to the ‘Design’ tab. Here, enter 800 as the minimum width:



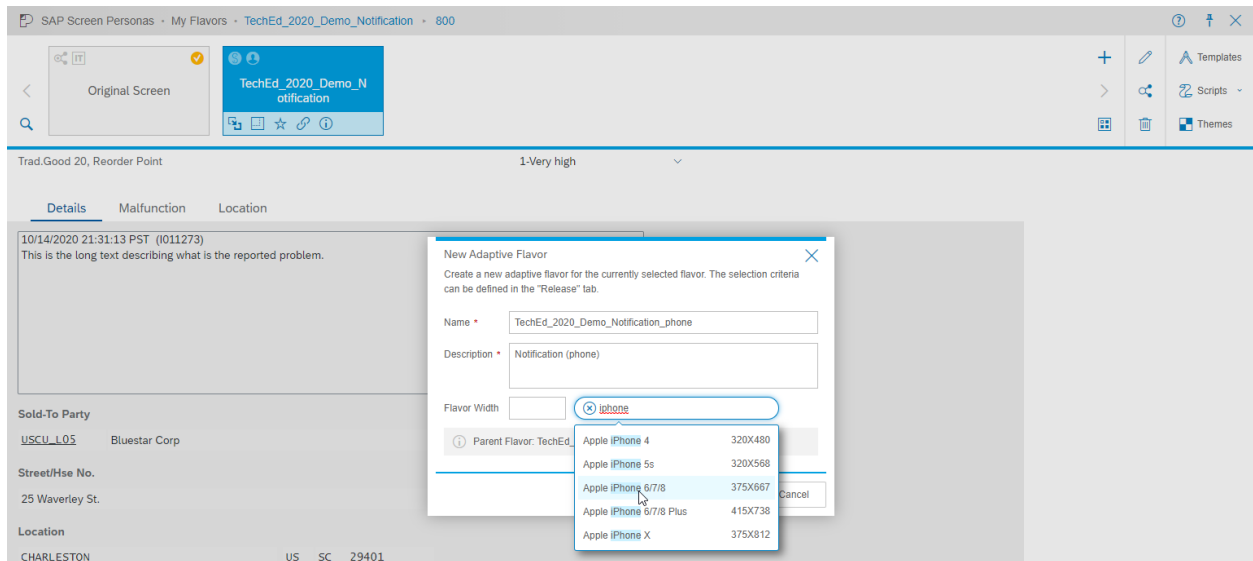
Save the change and exit the editor.

Click on the flavor tile, then select ‘Show Adaptive Flavors, then ‘Add Adaptive Flavor’.



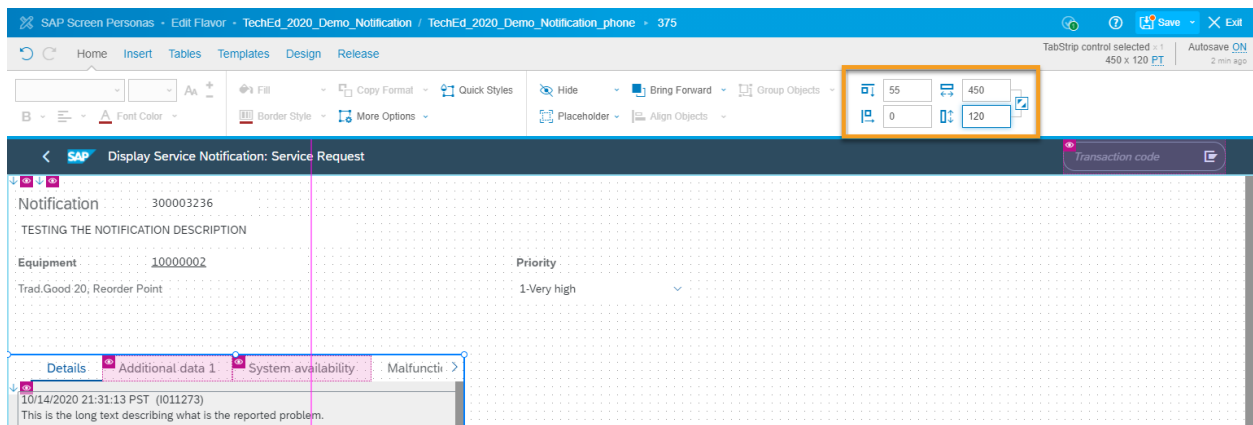
Remove the pre-selected width, enter ‘iphone’ in the device type field, and select iPhone 6/7/8 from the dropdown. Notice that the minimum width setting is also taken from this selection, so your new phone flavor is set up properly in the adaptive stack, and will be automatically selected if users runs the parent flavor on their device with a screen narrower than the parent flavor’s 800 width.



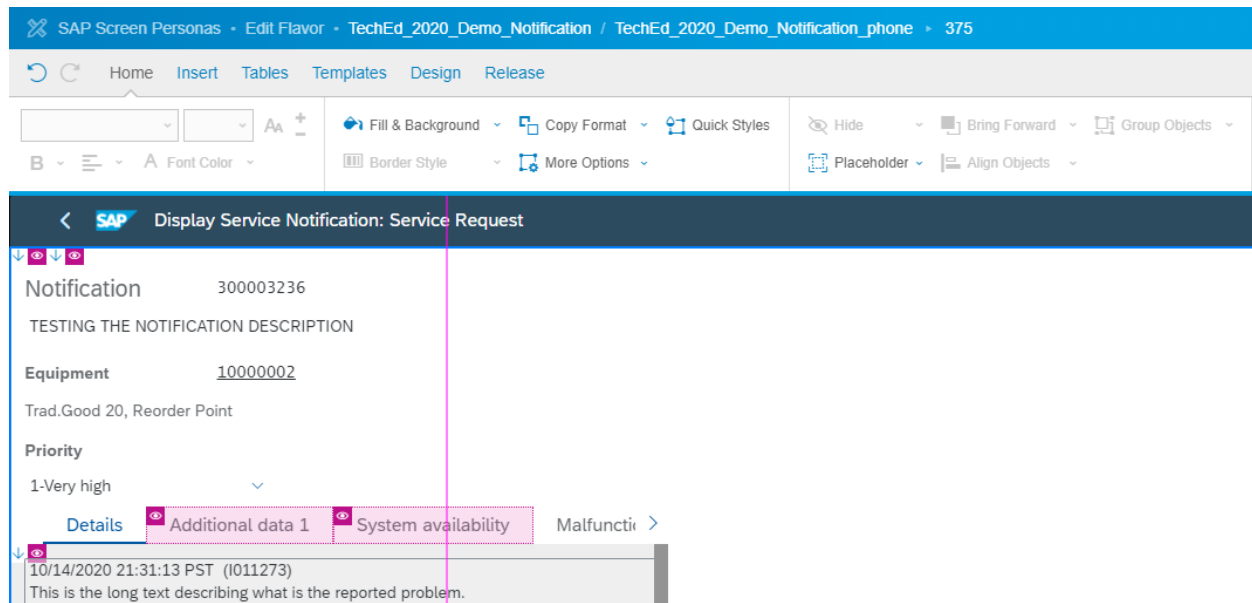


Your phone flavor inherits the settings from the parent tablet flavor. You need to rearrange the controls somewhat, so they fit the changed screen dimensions better.

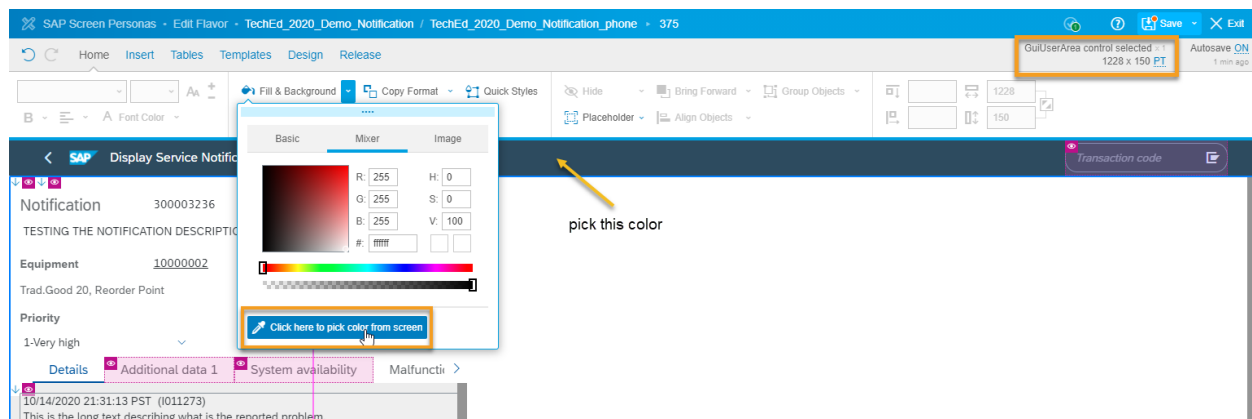
First, move the tab strip down so the Priority can fit above it, and make it narrower too. The recommended settings are:



Now, move the Priority to the left, under Equipment:



Let's change the header background color to match the standard title bar. Select the User Area by clicking outside the tab strip and not on any header field, then Fill & Background, Mixer and the color picker. Pick the title bar's color:



Then set the same fill color for all objects in the header, and their font color to white. The resulting layout should be like this:

< **SAP** Display Service Notification: Service Request

**Notification**

TESTING THE NOTIFICATION DESCRIPTION

**Equipment**

Trad.Good 20, Reorder Point

**Priority**

1-Very high

**Details** **Malfunction** **Location**


10/14/2020 21:31:13 PST (I011273)  
This is the long text describing what is the reported problem.

**Sold-To Party**

Bluestar Corp

**Street/Hse No.**

**Location**



Let's move on to the other two tabs. Do the same and rearrange the layout to get them look like this:

< **SAP** Display Service Notification: Service Request

**Notification**

TESTING THE NOTIFICATION DESCRIPTION

**Equipment**

Trad.Good 20, Reorder Point

**Priority**

1-Very high

**Details** **Malfunction** **Location**


**Start Date**

**Start Time**

**End Date**

**End Time**

☒ Breakdown



< **SAP** Display Service Notification: Service Request

**Notification**

TESTING THE NOTIFICATION DESCRIPTION

**Equipment**

Trad.Good 20, Reorder Point

**Priority**

1-Very high

**Details** **Malfunction** **Location**

**Functional loc.**

**MaintPlant**

Plant 1 US


**Location**

Compressor # 1

**Room**

**Company Code**

Company Code 1710



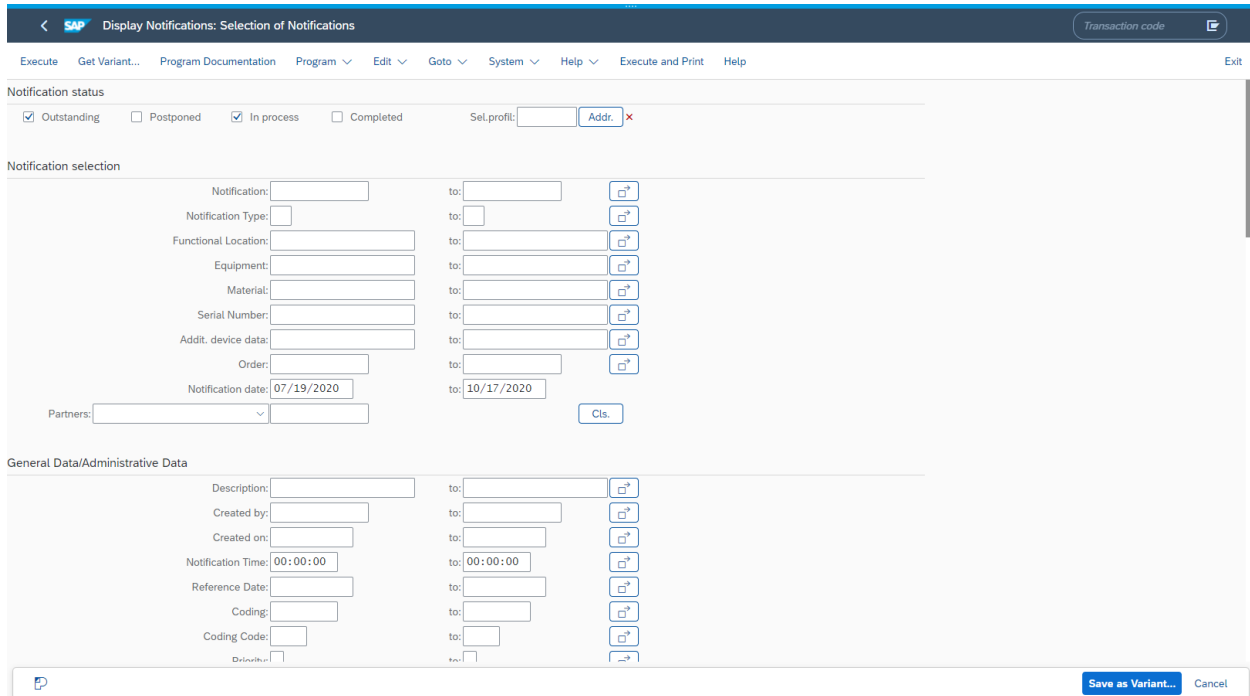


Now, you can test how the proper flavor is selected. Decrease the width of your browser window by dragging its vertical borders. Once you hit the defined width threshold, the flavor according to the current width will be picked.

You can test this on your phone too. Log on with your credentials via the same URL and run IW29, then select a notification from the list. Turning the screen orientation will adjust the layout according to the currently active width, both in the notification list and your notification flavor.

## **Fourth task: Adjust the selection screen of the Notification List**

The selection screen of transaction IW29 is very busy, with most selections irrelevant for our users:

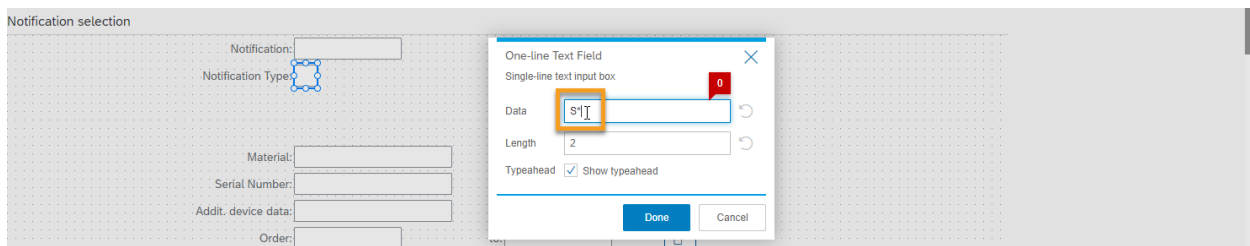


Besides, there are a few things we can improve in the process:

- Since we only want to consider the service notifications, it makes sense to automate that selection
- While the SAPUI5 table does offer filtering capabilities, the notification date range selection is not part of this and that may be important for our users to change from the default
- Besides selecting per equipment, we also want to restrict our list by Functional Location

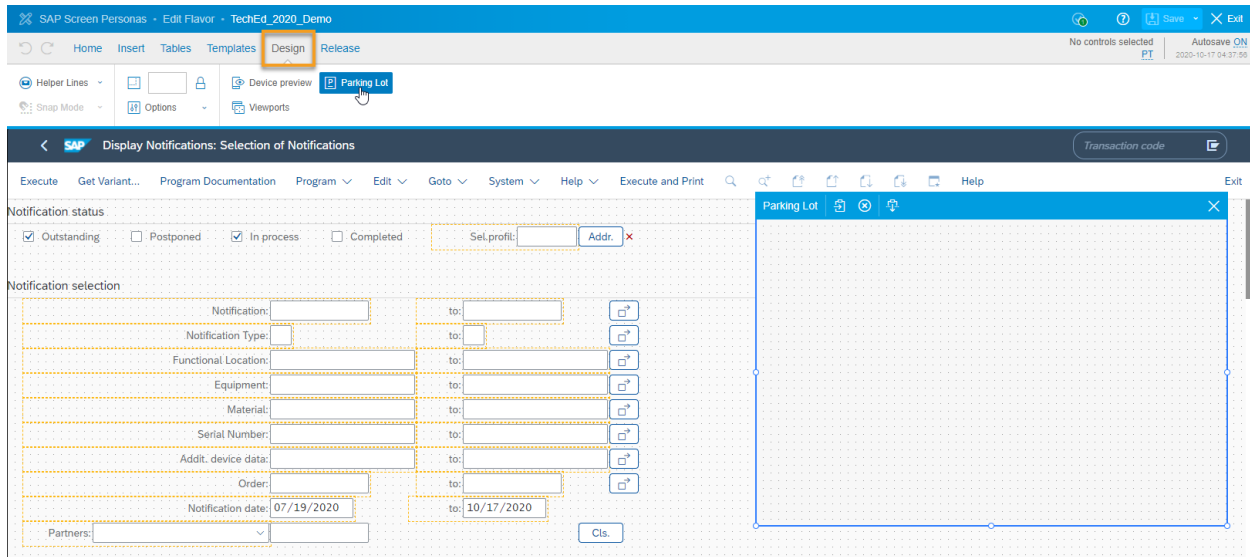
So, let's simplify it as the first step and remove everything we don't need.

First, set the Notification Type selection value by double-clicking the "from" value. This opens a popup where you can set the Notification Type value to S\*:



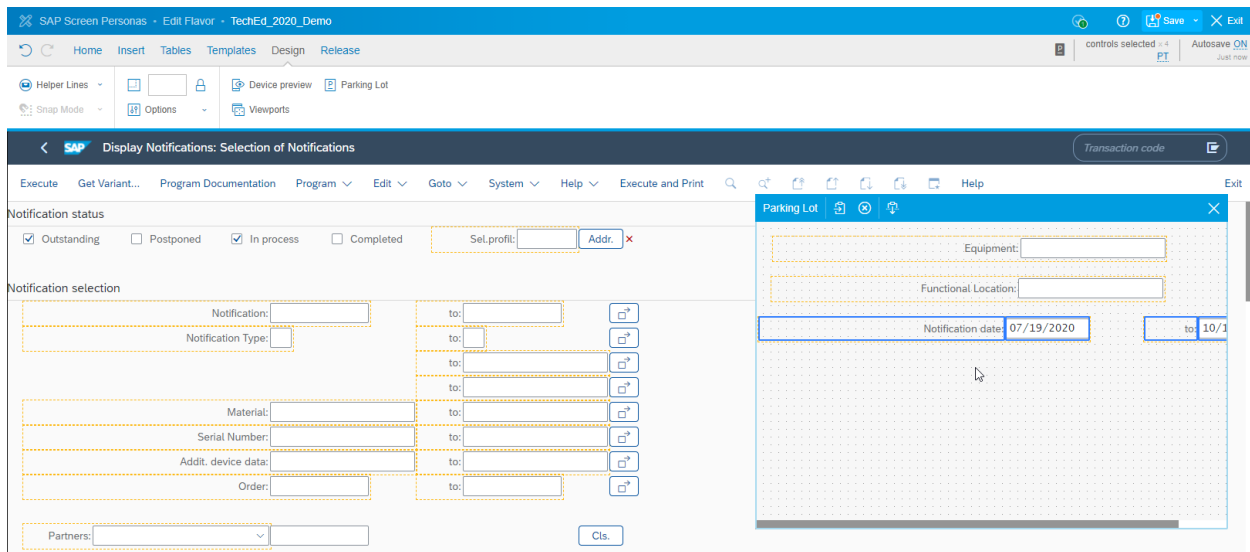
Hit 'Done', and now the selection is pre-set as part of the flavor.

Moving on to hide the unneeded objects, open the 'Parking Lot' in the Flavor Manager's 'Design' tab.

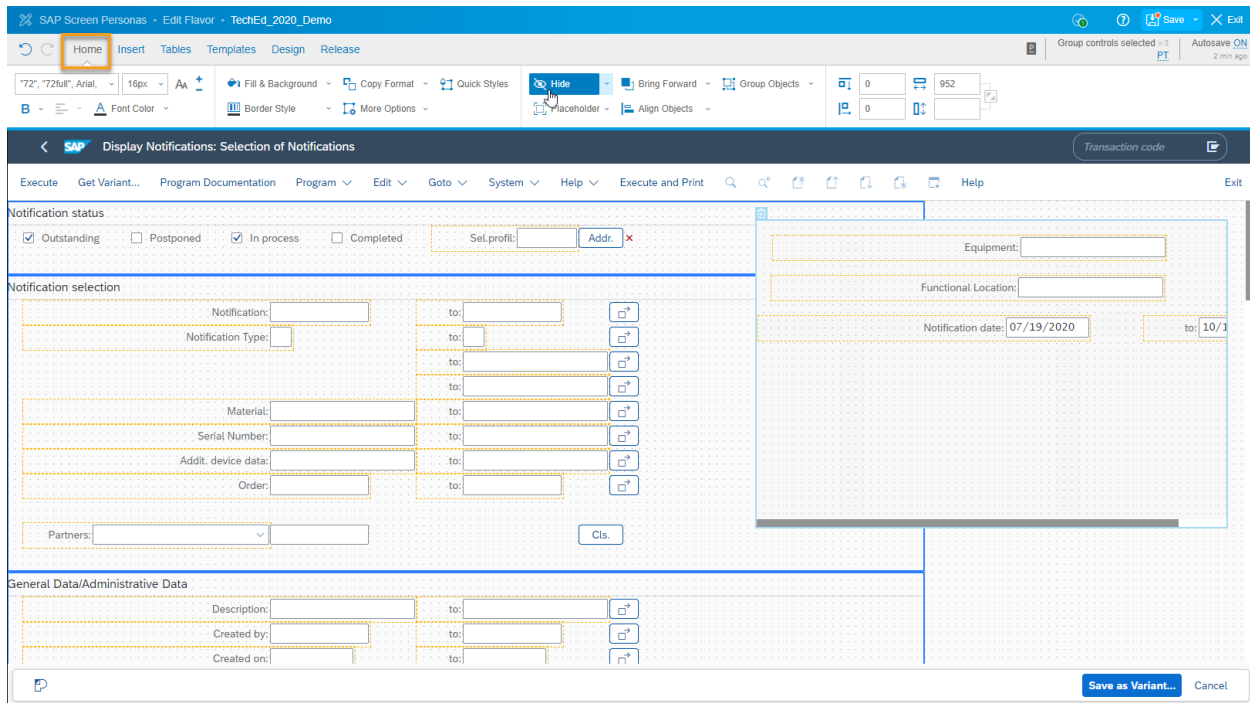


Let's move the labels and selection fields we want from the standard screen to the 'parking lot'. You can multi-select controls like explained earlier. When you selected the fields you need, simply drag them to the parking lot.

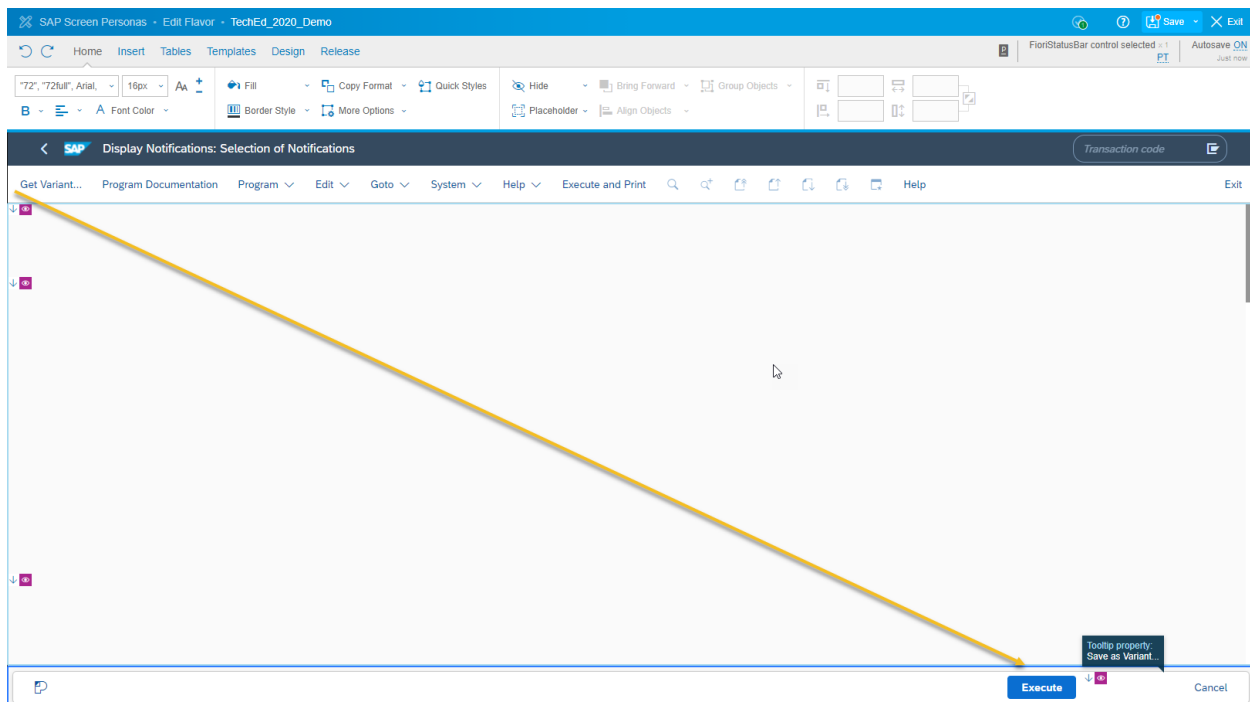
We only need the first ('From') selection for Equipment and Functional Location, and the notification date range.



Let's hide all other selections. Since they are grouped in boxes, we only need to hide those container group boxes. Multi-select them by the box label and hit the 'Hide' button in the Flavor Manager's 'Home' tab. Don't forget to scroll down and hide every remaining group box and selection field. The selection screen is quite tall.

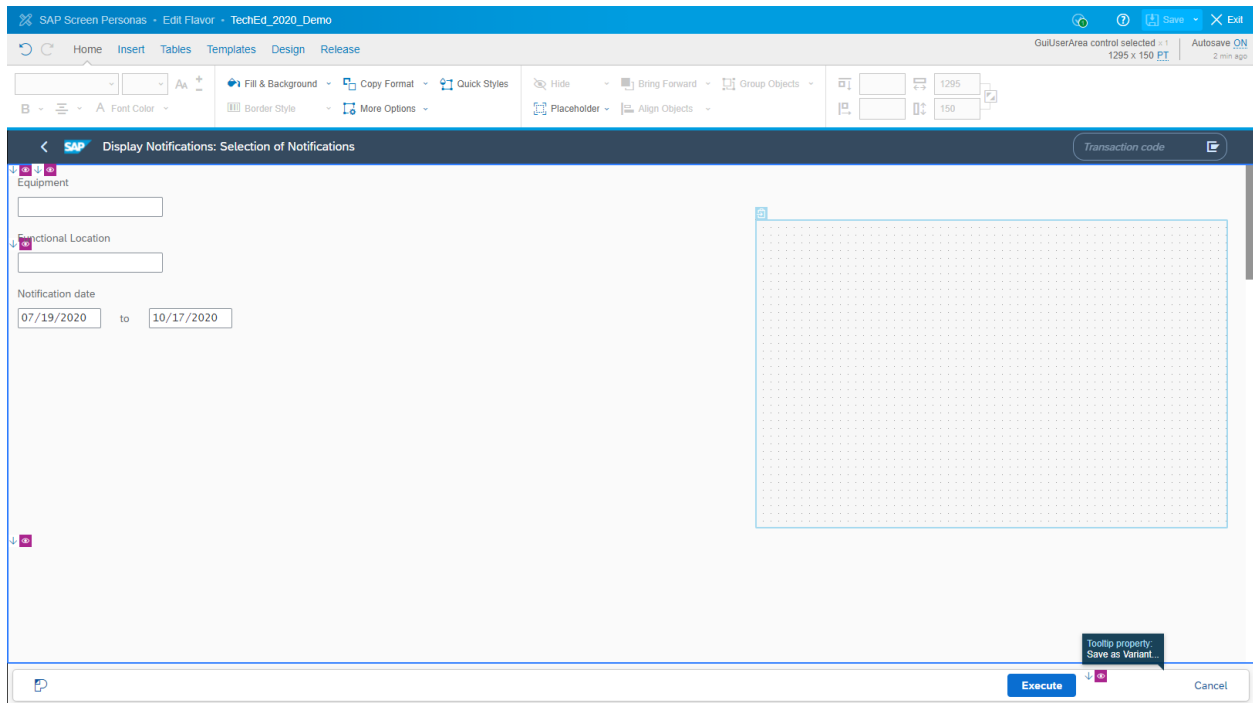


Now, drag the 'Execute' button of the application toolbar to the footer, next to 'Save as Variant', then hide 'Save as Variant'.



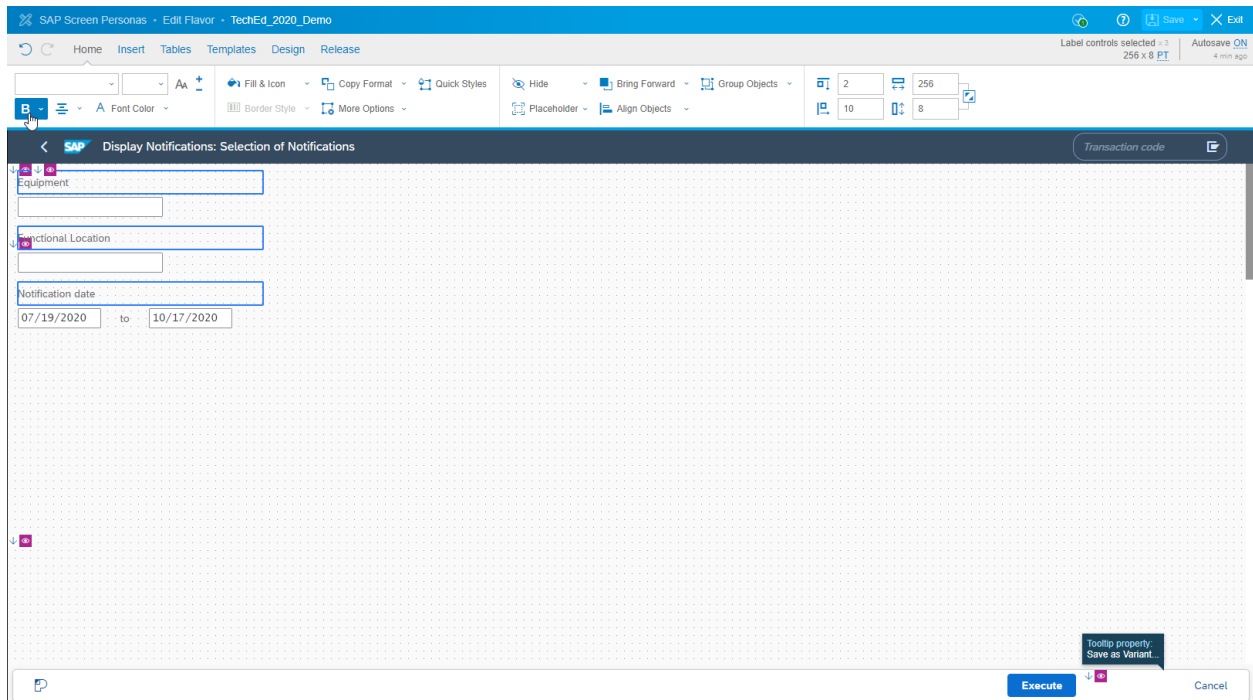
Hide the application's menu bar (at the top) as well, since we will not need other buttons from it.

Select the 'Design' tab in the Flavor Manager and bring back the 'Parking Lot' where the stored selection fields are. Start moving them to their final position, so the screen looks like this:



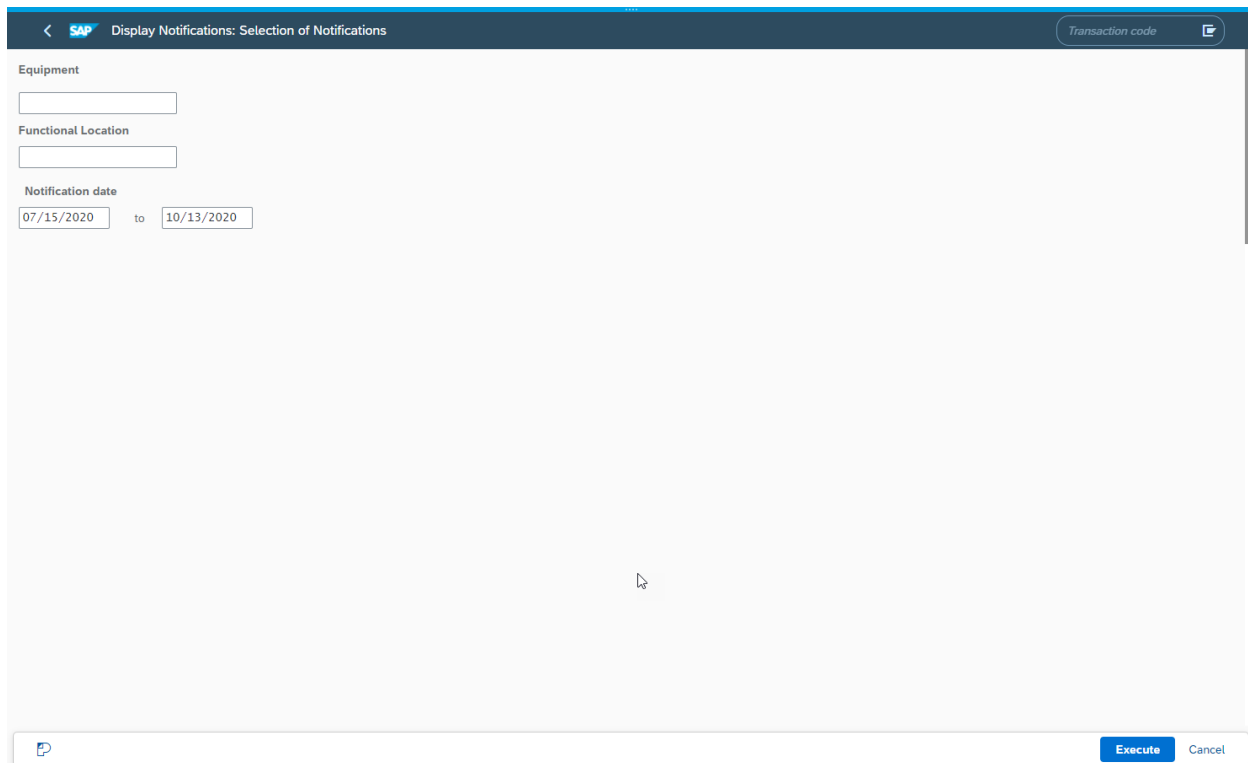
Close the 'Parking Lot'. Arrange the labels and input fields as shown. Remember that there are alignment tools for your disposal, to ease the process of lining up objects horizontally or vertically alike.

Select the three labels, make their font bold.





Now the selection screen is done:



Save and exit the Flavor Editor.

The only remaining issue is that if the user changes the selection values on this screen, the current 'Back' button script in transaction IW53 will not bring back the previous result list. This is because other than the notification type set in our flavor, all other selections will use the default values. So, we need to save the selection screen values and as part of the 'back' script in IW53, we need to reapply them when returning to IW29. Using the session store is the simple solution for this.

This means that the 'Execute' button function must be enhanced with storing the selections. In this case, we need to add something to the 'Execute' button but want to keep the standard button functionality too. Let's create the following script called *saveSelections*:

```
// Store selection values
session.utils.put('equipmentNr', session.findById("wnd[0]/usr/ctxtEQUNR-LOW").text);
session.utils.put('funcLoc', session.findById("wnd[0]/usr/ctxtSTRNO-LOW").text);
session.utils.put('dateFrom', session.findById("wnd[0]/usr/ctxtDATUV").text);
session.utils.put('dateTo', session.findById("wnd[0]/usr/ctxtDATUB").text);
```

Add this to the 'Execute' button's *onClick* event, like you did this earlier with the 'Back' button in transaction IW53. The difference is that in this case, first our added script runs upon the button click, but then we also let the standard 'Execute' function to continue, instead of suppressing it.

Finally, to use these saved selections, you need to add something to the 'back' script in IW53. So, go back to displaying a notification, and enhance your earlier script. The following snippet should go after starting transaction IW29, but before clicking 'Execute':

```
// Re-populate selection screen fields
session.findById("wnd[0]/usr/ctxtEQUNR-LOW").text = session.utils.get('equipmentNr');
session.findById("wnd[0]/usr/ctxtSTRNO-LOW").text = session.utils.get('funcLoc');
session.findById("wnd[0]/usr/ctxtDATUV").text = session.utils.get('dateFrom');
session.findById("wnd[0]/usr/ctxtDATUB").text = session.utils.get('dateTo');
```

Save your updated script and test if everything works the way it should. Go back to the IW29 selection screen and enter a different date range than the default, or equipment or functional location you know exists. After navigating to the notification list and displaying a notification, try how the 'Back' button works. The expectation is that you will be taken back to the exact same list you had earlier, before selecting a notification.