



POPULUS

LUXOFT POPULUS TRAINING

CD Player Tutorial

Table of Contents

1	INTRODUCTION	3
2	POPULUS PROJECT	4
2.1	EXERCISE A2 – CREATE A HELLO WORLD HMI	4
3	FU INTERFACES	7
3.1	EXERCISE A3.1 – DESIGN AN FU INTERFACE	7
3.2	EXERCISE A3.2 – CREATE AN FU INTERFACE	8
4	HMI CREATION BASICS	10
4.1	EXERCISE A4.1 – CREATE A CD PLAYER HMI	11
4.2	EXERCISE A4.2 – ADD BUTTONS AND TRACK INFO	15
5	LANGUAGES – EXERCISES	20
5.1	EXERCISE A5.1 – CREATE A TEXT MODULE	20
5.2	EXERCISE A5.2 – TRANSLATION	21
6	FACEPLATE – EXERCISES	24
6.1	EXERCISE BC6 – CREATE A FACEPLATE	24
7	HMI CREATION DETAILED – EXERCISES	27
7.1	EXERCISE C7.1 – ADD HMI LOGIC	27
7.2	EXERCISE C7.2 – ADD A SETTINGS PROFILE	30
7.3	EXERCISE C7.3 – ADD PROFILE TRANSITION EFFECT	35
7.4	EXERCISE C7.4 – ADD BUTTON ANIMATIONS	37
8	RUNTIME	40
8.1	EXERCISE BC8 – SIMULATE A FUNCTIONAL UNIT	40
9	[DEPRECATED]	41
10	HMI CREATION ADVANCED	42
10.1	EXERCISE C10.1 – ADD A POPUP	42
10.2	EXERCISE C10.2 – ADD A SETTINGS MENU	46
10.3	EXERCISE C10.3 – ADD ANIMATION FIELD	51
10.4	EXERCISE C10.4 – ADD A SCROLLBAR	53
10.5	EXERCISE C10.5 – ADD A TRACK LIST	55
10.6	EXERCISE C10.6 – ADD THE LANGUAGE MENU	61
10.7	EXERCISE C10.7 – PRIORITY STACK	67
11	FU IMPLEMENTATION	68
11.1	EXERCISE B11.1 – SIMPLE CD PLAYER STUB	68
11.2	EXERCISE B11.2 – IMPLEMENT SUPPORT FOR COVER ART	74
11.3	EXERCISE B11.3 – IMPLEMENT TRACK LIST	77

1 INTRODUCTION

This tutorial aims to familiarize you with Populus Editor. After finishing the exercises you will be able to implement your own projects in Populus.

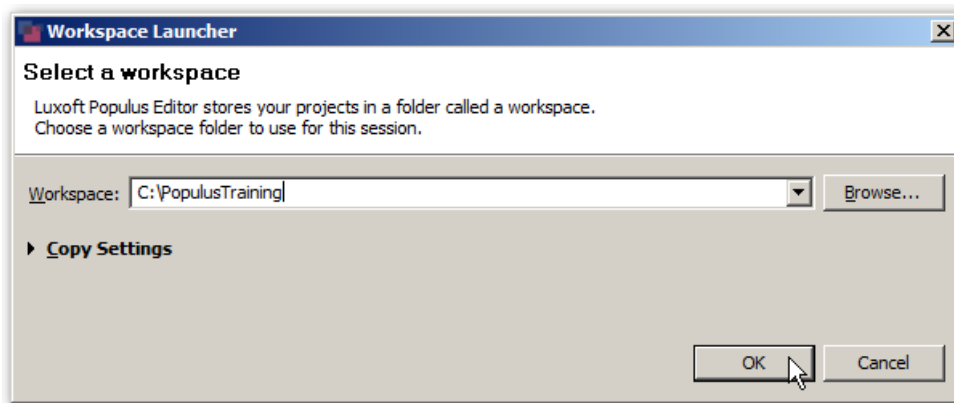
2 POPULUS PROJECT

2.1 EXERCISE A2 – CREATE A HELLO WORLD HMI

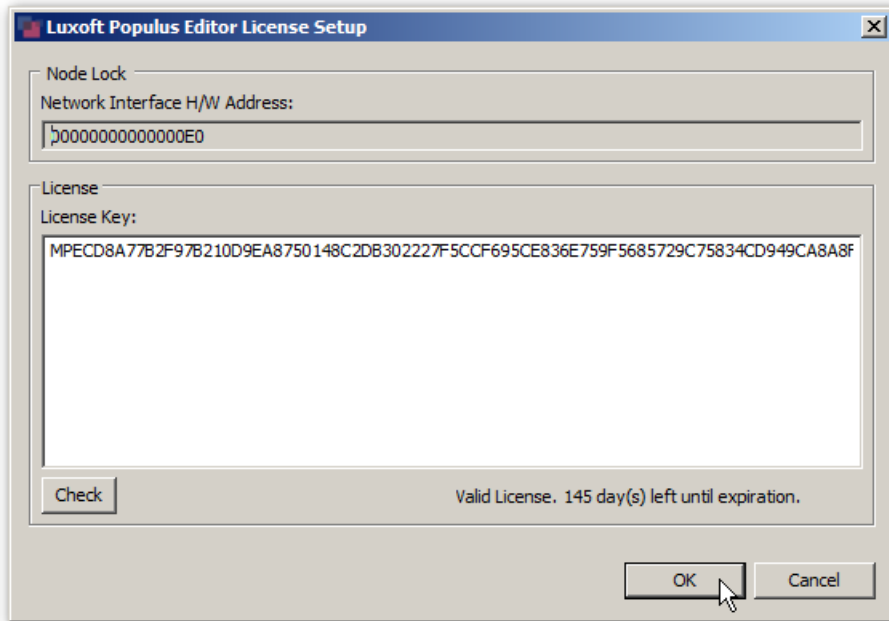
In this first exercise, you will start to get familiar with the Luxoft Populus Editor. To do that, we first need to install the Editor from the USB stick that you received with these exercises.

1. Insert the USB memory stick into your computer and navigate to it.
2. Run the installer exe to install both Luxoft Populus Editor and the Windows version of Luxoft Populus Engine.
3. After having started the Editor for the first time, set up a workspace folder where you will store your projects. If you have already started the Editor for the first time, you can still switch workspace folders by going to the *File* menu and choosing *Switch Workspace*. Click *Other* or choose a recently used workspace.

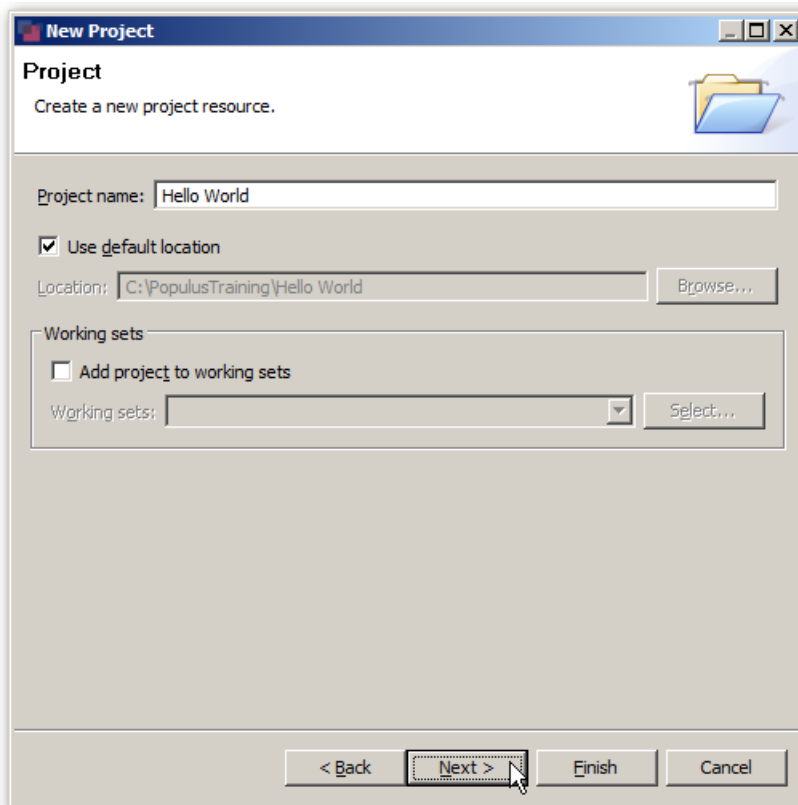
Example : C:\PopulusTraining\



- Next, go to *File > Setup License...* to set up your license key for the Training. You will find the license key for the training on the USB Stick as well, under the folder "License". Open the text file and copy & paste the contents into the *License Setup* dialog.



- To create a sample Hello World project, go to *File > New > Other...* and select to create a new *Populus Project*.



6. Select to generate sample content, and give the individual files suitable names if the prefilled ones are not to your liking.
7. After having clicked on *Finish* your project has been set up and the generated Deployment file has been set as the current.
8. To try out this HMI in the Engine, go to the *Target* menu and click on *Serialize* (or press F7).
9. A generated project with no modifications made to it should serialize without any errors, so all steps should be green or marked with "Skipped". Click *OK* afterwards. The serialization is now complete.
10. With a successful serialization, run this HMI in the Engine by clicking *Run* in the *Target* menu (or press F10).



11. (*Optional*) Go back to the Editor and open the "DDH\(\Master) Hello World.ddh" file. Feel free to look around and do modifications. To try out your changes, serialize again and restart the Engine by clicking *Serialize and Run* in the *Target* menu, or pressing F9.

3 FU INTERFACES

3.1 EXERCISE A3.1 – DESIGN AN FU INTERFACE

In the next exercise, we will create an FU interface for a simple CD player FU. It will be used by your HMI later on in this course.

1. Think for a minute or two about what kinds of Events, Actions, Indications and Data you would expect a CD player FU to have.

3.2 EXERCISE A3.2 – CREATE AN FU INTERFACE

It is time to fire up the Editor and actually create the FU interface there. Use the “Hello World” project created in Exercise A2 and create the FU Interface in its FU folder.

1. Create the FU interface according to the description below (*File > New > Other... > Luxoft Populus > Functional Unit Interface (FUClass)...*).

Your CD player FU (DemoCDFU) will have the following interface:

- FU ID = 192;
- Events:
 - CD Read Error;
- Actions:
 - Play, Pause, Next, Previous as simple actions;
 - Volume Up, Volume Down as simple actions;
 - SelectTrack as value action taking an Integer;
- Indications:
 - CD Present;
 - Next Track Available;
 - Previous Track Available;
 - Playing;
- Enumeration Sets:
 - CD State as a set of different states (e.g. Playing, Loaded, No CD);
- Dynamic Data:
 - Number of tracks as Integer;
 - Track Time Left as Time;
 - Track Total Time as Time;
 - Track Time Elapsed as Time;
 - Track Number as Integer;
 - Track Artist as String;
 - Track Title as String;
 - CD State as an Enumeration Value of CD State;
 - Track Cover Art as Bitmap;
 - Track List as List, where each item of the list has at least the track cover art (Bitmap) and the track title.

2. Lock all IDs to a new version 1.0
 - a. Go to the *Version* tab and fill in the details for the new version;
 - b. Press *Release Version* button to the left;
 - c. Verify that all IDs are locked;
 - d. Verify that IDs can't be removed once the lock is in effect;
 - e. Verify that it is still possible to add new IDs, i.e. new data for Volume as Integer.

4 HMI CREATION BASICS

We are going to create a CD player HMI. For this, we have a pre-built simulation of a CD player that plays mp3 files. Its FU executable is already built, which means you need to have the exact same FU interface that it was built with.

All you need to get started with the CD Player is bundled with the install kit. To get these projects into your current workspace, follow the instructions below:

1. Click on *File > Import*. Choose *General > Existing Projects into Workspace*.
2. *Select root directory* by browsing to:

```
C:\Program Files (x86)\Luxoft Populus xx\Exercises\HMI\Education  
Exercises
```

Note: beginning with release 4.46.0 examples and exercises can be installed to a user-defined directory. The default location is `c:\Users\<user name>\Luxoft Populus xx`.

3. Make sure the option *Copy projects to workspace* is checked, so that you get copies of all exercises to work with.
4. Import all the “CD Player xx” projects.

The project called “CD Player Common” contains all pre-made material that we will use in the following exercises. Bitmaps, Text Modules, FU Interfaces and other miscellaneous files.

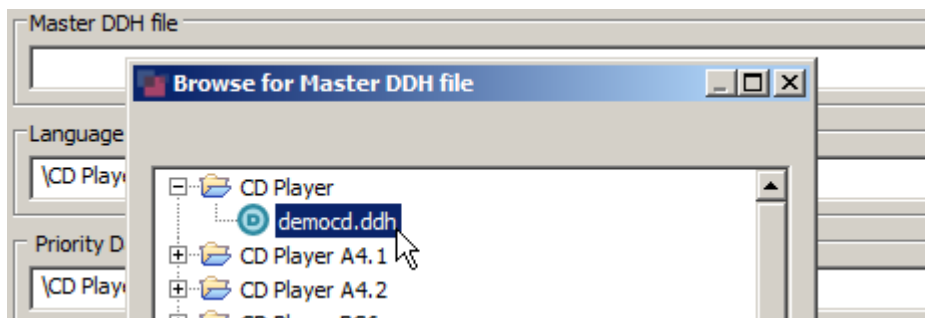
There is also a project called “CD Player” that we will use to create the CD Player HMI. Its Deployment file references files within the “CD Player Common” project. The reason for this split is mostly to make it possible to jump ahead to a different exercise if for instance there is a lack of time to finish it.

From here, you can jump to any of the exercises and start off at that point by opening the project that corresponds to the previous exercise. The final result of each exercise is stored in a separate CD Player project. E.g. to start with Exercise C10.1, open the “CD Player C07.4”.

4.1 EXERCISE A4.1 – CREATE A CD PLAYER HMI

So, let's start with the creation of a CD Player HMI.

1. Find the "CD Player" project and set the "CDPlayer.deploy" as the active deployment.
2. Create a new DDH file in the folder "CD Player" and call it "democd.ddh". Do not create default contents in it.
3. Open the Deployment file (by double-clicking it) and set the Master DDH to be the file created in the previous step. Save and close the Deployment file.



4. In the democd.ddh (must be the **Master DDH!**), go to the *Master Settings* tab, *General* sub-tab and set up the following:
 - a. Make the display 480 x 320;
 - b. Make current language to be controlled by the FU called "Settings" and the Dynamic Data ID "DisplayLanguage". To assign current language, click the corresponding *Set* button.
5. In the visual model, descend into Profiles, and create a Profile called "CD".
6. Create a new page within this profile, by right-clicking it and selecting "Add New Page to Page Traversal Order". Call this page "CD Player", and click it inside the profile.
7. Edit the page (Press *EDIT* in the page state).
8. Set the background bitmap of the page. For this, we must setup the Bitmap ID for the background first:
 - a. Go to the *Bitmaps* tab;
 - b. Press *Add...*;
 - c. Select `\CD Player Common\Bitmaps\bg.png` and press *OK*;
 - d. Back in Visual Model Editor, either right-click the page and choose *Set Background Bitmap...* or select the Page and set the background bitmap within the *Details* pane (hidden behind the *Project Explorer* by default).
 - e. Create a new Panel on the Page (right-click and select *Create New Panel...*). Call it "CD Title".



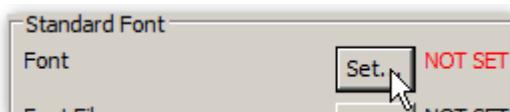
9. Adjust the first panel so that we can use it for the title of our CD Player:
 - a. Move and resize it so that it covers the top third of the display.
10. Add *StaticTextFields* to it so that it looks like this:



- a. First we need to add a Font ID so that it is possible to draw text. Go to the *Fonts* tab, click on *Add...* and call it "Normal".
 - b. Double-click the font value (currently marked "NOT SET") to open the *Edit Font* dialog.

I.	Name	Doc	Default (Active Skin)
	Normal	...	NOT SET

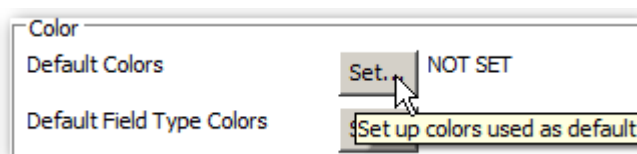
- c. Make sure the Font Type is Standard Font.



- d. Click the *Set...* button and set the font to *Arial, Narrow, 14 pt.*
 - e. Add a new Font ID, called “Title large” set to *Arial, Narrow Bold, 28 pt.* (Use Copy & Paste if you like.)
 - f. Also add a third font, called “Title small” set to *Arial, Narrow, 12 pt.*
 - g. Go back to the Visual Model and add a *StaticTextField* by right-clicking the panel and selecting *Create New Field....* Call it “Title Large”.
 - h. Set its Text ID to the text in your Text Module that holds this title.
 1. In the *Details* panel on the left find *Static Text Field* group and click *SET...* for *Static Text ID*;
 2. Find the Text Module called “CDPlayer”;
 3. Select the “CDPlayer_Title” text entry.
 - i. Set the Font to “Title large” created in the previous steps.
 - j. Adjust the size and position of the field.
 - k. Add another *StaticTextField* as above for “Luxoft Populus Training” but use the “Title Small” font instead. Call it “Title small”.
 1. For *StaticTextID*, use the “CDPlayer_Subtitle” text entry in “CDPlayer”.
11. It’s apparent that we need to define colors as well (cyan/red is shown to indicate that no colors are set). Set up two Colors, one called “Transparent” and one called “Text Color”. Set the *Alpha Blend* value of the first one to 0 to make it fully transparent, and set the text color to *white* with a slight amount of *Alpha Blend* (200).
- a. Go to the *Colors* tab and add two new Colors.
 - b. First double-click in the *Default (Active Skin)* column for “Transparent” to open the *Set Color Properties* dialog to set its color. Make sure that the Alpha blend value is set to 0, as this will make the color transparent.
 - c. Do the same with “Text Color”. Set the RGBA values to (255, 255, 255, 200); this will create a white font that is slightly transparent.

Notice how the name of the Color does not correlate to the actual color value, but to the function the color has in the HMI. By sticking to such naming scheme, it is easier to create new skins or themes in the future.



12. Go back to the text fields and select one of them to set up the default field color (to be used for all fields throughout the system).
- a. In the Visual Model Editor, select the “Title Large” static text field.
 - b. Go to field details and setup Default Field Colors; this will affect all fields – and we would like all fields to have a reasonable default setup.





- c. Set "Text Color" as *Default Foreground Color* and "Transparent" as *Default Outline Color* (= Background).



13. To make the HMI work, we also need at least one Frame. Go to the *Frames/Profiles* tab and create a *Frame* (call it "Default"); set the *Default Profile ID* to "CD".

Frames						
Name	Default Profile ID	Control Map	Event Map	State Chart	History Stack Si...	Doc
 Default	 CD	NOT SET	NOT SET	NOT SET	NOT SET	...

14. We also need to set the Parent Frame of Profile "CD" to the Default frame.

Profiles						
Name	Parent Frame ID	Control Map	Event Map	State Chart	Doc	
 CD	 Default	NOT SET	NOT SET	NOT SET	...	

15. Make sure that *CDPlayer.build* is set as the active build file (it should be displayed in italics). If not, right-click the file and select *Set as Active Build File*.

16. Serialize and run the HMI in Populus Engine

- a. From the *Target* menu, select *Serialize...*
- b. Correct any remaining verification errors (needs to rerun serialization after correction).
- c. From the *Target* menu, select *Run Preferences...*
- d. Change the configuration file to be the `default.cfg` file found in the "CD Player" project.
- e. Run the HMI (*Target > Run*).
- f. Start the CD FU executable (to run the simulated FU) if that is not running. You can find it in
C:\Program Files\Luxoft Populus xx\Exercises\FU\FUCD.exe
- g. Start the Settings FU executable, if it is not running:
C:\Program Files\Luxoft Populus xx\Exercises\FU\FUSettings.exe

Note: beginning with release 4.46.0 examples and exercises can be installed to a user-defined directory. The default location is `c:\Users\<user name>\Luxoft Populus xx`.

4.2 EXERCISE A4.2 – ADD BUTTONS AND TRACK INFO

The CD Player HMI is going to be extended with buttons and a pane with some information about the currently playing track. This will both control the FU (Play, pause etc.) and set up subscriptions for data to show the info.

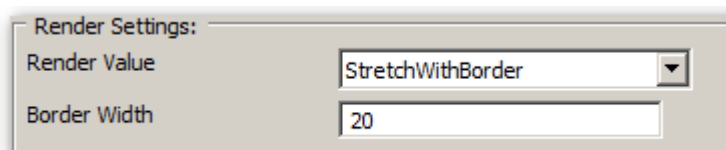
We will begin with the buttons.

1. In the “CD Player” page, create a new panel called “Buttons”.
2. Position the panel in the bottom of the screen. Make it about 70 pixels tall, which will leave some room for the information above the buttons

This is the end result we have in mind:



3. The idea is to start with the “Previous track” button, and make four copy/paste’s of it. Right-click in the panel and select *Create New Field...*
4. Select *Active Control Field*. Call it “Previous”.
5. Adjust the field position and size so that five buttons will fit (see image above).
6. To decorate the button, we will use a common background bitmap and apply a separate icon for each of them. Go to the *Bitmaps* tab to add a few more bitmaps.
 - a. Click *Add...* to add “\CD Player Common\Bitmaps\button-normal.png”.
 - b. We would like to decide the size of the button in the editor, so we are going to utilize the *Stretch With Border* feature. Change the *Render Settings* to *Stretch With Border* and set the border width to 20 pixels. Then click *OK* to close the dialog window.

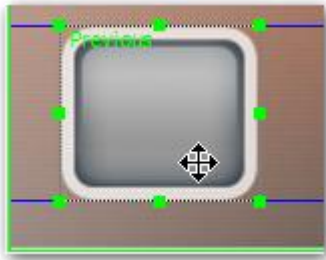


- c. We are going to use different bitmaps depending on the state of the button. Double-click in the *Active Skin* column of the *button-normal* row to open up the *Set Bitmap Definition* dialog.
- d. For *Focused Bitmap*, set the bitmap “button-active.png”.
- e. For *Selected Bitmap*, set the bitmap “button-selected.png”.
- f. For *Disabled Bitmap*, set the bitmap “button-disabled.png”. Note that for all three additional states the *Stretch With Border* render type and the border width of 20 pixels are set.
- g. Done. Click *OK*.

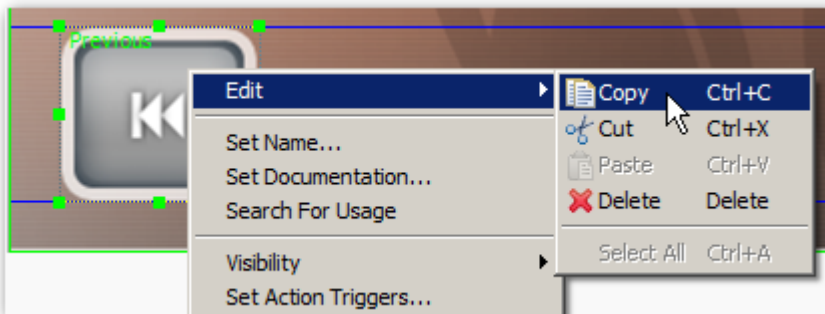
7. Next add the icons, five of them, using “Add ...” and selecting all the five icon bitmap files: “NextIcon.png”, “PauseIcon.png”, “PlayIcon.png”, “PrevIcon.png”, “SettingsIcon.png”.

Note: the dialog allows multi-select to add more than one bitmap at once.

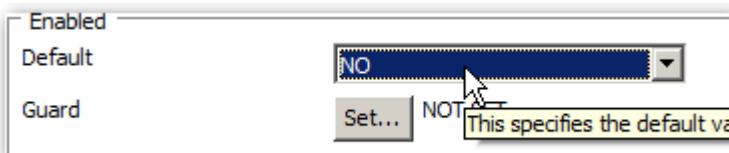
8. Back in the Visual Model Editor and with the “Previous” field selected, set its *Outline Bitmap* (in the *Field* section) to the normal button bitmap.



9. Set its *Select Action* (in the *Active Control Field Basic Properties* section) to send the External Action “Previous” to the CD Player.
10. Scroll to *Field Bitmap Properties* and enable bitmap, then set its icon as field bitmap.
11. Now, copy this field by right-clicking it in the editor area and selecting *Edit > Copy* (Ctrl-C).



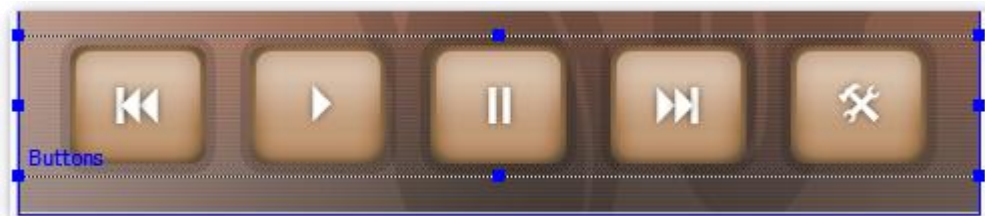
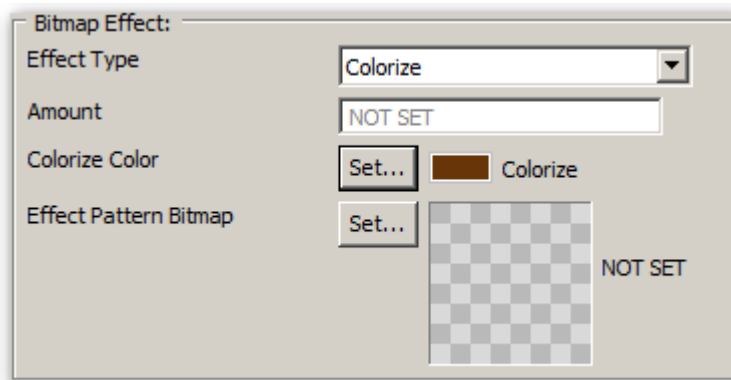
12. Right click again and select *Paste* (Ctrl-V). Make this the “Play” button.
13. Do the appropriate changes (change its *Select Action* to “Play” and change its *Field Bitmap* to the play icon).
14. Repeat steps 12-13 to create the remaining three buttons “Pause”, “Next”, and “Settings”. Do not set any select actions for the “Settings” button. At this point, we can even go ahead and disable it, set Enabled Default to false.



Feel free to serialize, correct all verification errors and run this in the Engine to make sure the CD is playing when you press “Play” etc. To make FUCD.exe detect mp3 files, run it as

FUCD.exe FUCD.cfg

15. It is possible to make the buttons harmonize with the rest of the HMI by applying a *Colorize Bitmap Effect* to the Bitmap ID for the button background.
- Setup a new Color called "Colorize";
 - Set its RGBA values to some brownish color;
 - Go to the Bitmap definition for the button bitmaps and double-click in the *Default (Active Skin)* column for "button-normal" bitmap;
 - In the dialog, press *Set...* for the "Default Bitmap".
 - In the dialog that opens, select the "Colorize" *Effect Type* and click *Set...* to select the "Colorize" color we just created.
 - Repeat steps d—e for other bitmap states (Normal, Focused, Selected, Disabled).
 - Done. Click *OK*.



16. Next, we will create the track information panel. It will contain some static text strings working as labels for the dynamic information, and dynamic text fields containing the actual information (Track number, Artist and Title).



17. Create yet another panel, called "Info".
18. We are going to assign it a background bitmap.
 - a. Go to the *Bitmaps* tab and add the file "LightSemitransparentBackgro.png". Set it to *StretchWithBorder* with a border of 10 pixels.
 - b. Use the newly added bitmap as background in the panel.
 - c. Resize and position the panel according to the image above.
19. Start with the static texts by adding a new *StaticTextField*, call it "Title".
20. Adjust its size and position according to "Title" in the image.
21. Set its *Text ID* to the text entry showing "Title:"
22. Set it to right alignment and use the font "Title small".
23. Assign it a new color that is more dimmed white:
 - a. Go to the *Colors* tab;
 - b. Create a new *Color* called "Dimmed text color";
 - c. Set its color to white with Alpha blend value of 100;
 - d. Set *Field Colors* for this field so that "Dimmed text color" is its foreground color and "Transparent" as its outline color (background color). (Make sure to not change *Default Colors* as this will change the colors for ALL fields, not just this field type.)
24. Make a copy/paste and create the "Artist" field.
25. With the two static fields ready, create a *DynamicTextField* for the track number called "Track Number".
26. Adjust its size and position according to above
27. Set Dynamic Data to "Track Number (integer)" in the DemoCDFU.
28. Set its Text – Default Text to the entry that contains format string "Track %##". %## is a placeholder for integer numbers.
29. Create the *DynamicTextFields* for title and artist in the same manner. This time, use a format string that says %STR%, which is a placeholder for a string.
30. Finally, create a *DynamicBitmapField* for cover art, 90 x 90 pixels.
 - a. Set its *Dynamic Data* to the "CurrentCoverArt" item in the DemoCDFU.
 - b. Set *Invalid Bitmap* to use the "NoCoverArtIcon.png" file by adding it in the *Bitmaps* tab first.
 - c. Set *Use Default Data for Rotation Angle* field. Select value type *Integer* and fill in the value of 0.
31. Set up runtime data so that the HMI can be previewed:
 - a. In the Project Explorer, create a new DDH Runtime file;
 - b. Set it as the active and open it;
 - c. Find the "TrackNumber" data in the CD FU, and give it a value. Find the "Title" and "Artist" strings and fill in text strings there as well;

- d. Go to *Settings > Preferences...* and select the *DDH Editor > Preview* tab;
 - e. Make sure *Show format strings instead of DDH Runtime data* is unchecked so that data can be previewed.
32. Serialize (correct any verification error and serialize again) and try the HMI in the Engine by running it.

Here is a recap of how to get it running (You could also use F9 to re-serialize and run the selected deployment):

- a. From the *Target* menu, select *Serialize...*
- b. Correct any remaining verification errors (needs to rerun serialization after correction).
- c. From the *Target* menu, select *Run Preferences...*
- d. Change the configuration file to be the `default.cfg` file found in the “CD Player” project.
- e. Run the HMI (*Target > Run*).
- f. Start the CD FU executable (to run the simulated FU) if that is not running. You can find it in
`C:\Program Files\Luxoft Populus xx\Exercises\FU\FUCD.exe`
- g. Start the Settings FU executable, if it is not running:
`C:\Program Files\Luxoft Populus xx\Exercises\FU\FUSettings.exe`

Note: beginning with release 4.46.0 examples and exercises can be installed to a user-defined directory. The default location is `c:\Users\<user name>\Luxoft Populus xx`.

5 LANGUAGES – EXERCISES

5.1 EXERCISE A5.1 – CREATE A TEXT MODULE

In this exercise, you will create a text module for your system. This text module will contain the text strings required for a simple CD Player home screen.

In your “Hello World” project created in Exercise A2, there is a folder called “TMod”. A Language Definition file (*.langdef*) is already provided which contains English and Swedish, with English as the default, alongside a ‘Hello World’ text module.

1. Double-click the langdef file to review it. Optionally change “Swedish” into a language of your liking that you know. (You don’t really need to know the language to complete this exercise.)
2. Create a new Text Module for a CD Player HMI, called CDPlayer.tmod. (*File > New > Other > Text Module*). Give it module ID 2.
3. Make sure that the deployment in the Hello World project is active and add the new Text Module to the Deployment by right-clicking it and selecting “Add to Active Deployment”.
4. Create language modules for English and for the other language to translate to by pressing the *Add Language Module* button. (It doesn’t matter if you know that language or not – just make up the translations when we come to that step).
5. Don’t forget to add a default variant (Variant 1) in the *Variants* tab – review: What are variants used for?
6. Add the entries required for this screen: (Note that some are dynamic data strings originating from the DemoCDFU).



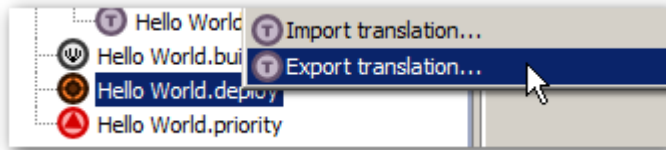
7. Now, go to the English tab and fill in the text strings for English.

5.2 EXERCISE A5.2 – TRANSLATION

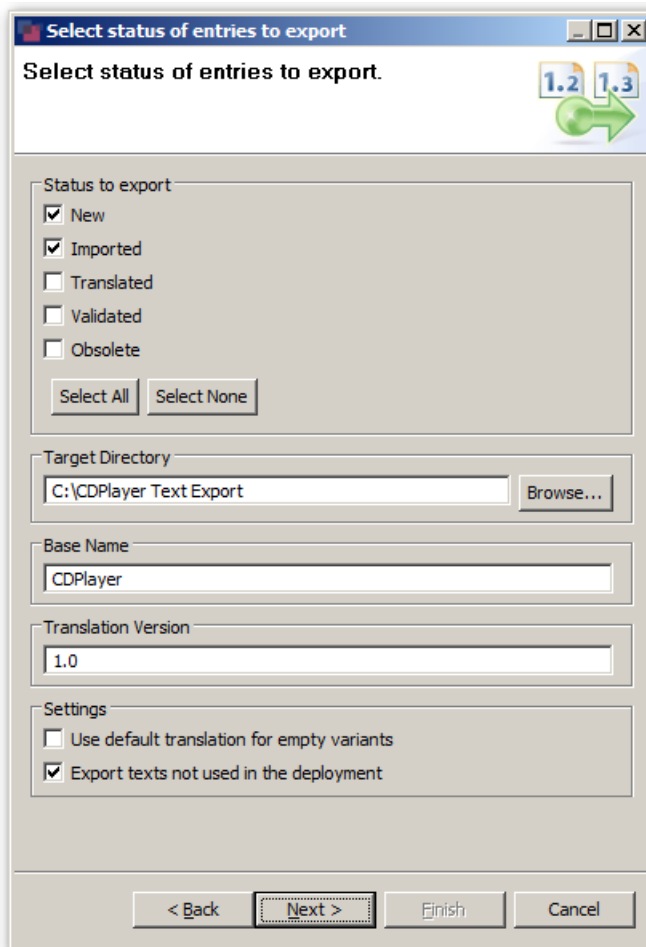
With the Text Module created in the previous exercise, simulate sending it to a translation bureau by exporting the entries you want translated.

Perform the translation in the XML file (.trans file) and import it back into the Text Module.

1. Set up the status of all entries for the target language so that you don't export entries that you are not interested in having translated. The status of an entry is set by right clicking it in the .tmod file in the tab for the language for which you want to set the status.
2. Export the translation (.trans) files for all languages:
 - a. Right-click the "Hello World" deployment file and select *Export translation...*



- b. Select all languages and Text Modules and press *Next*.
 - c. Select appropriate export statuses and make sure that the setting *Export texts not used in the deployment* is selected. Make sure to select existing folder for the *Target Directory*.

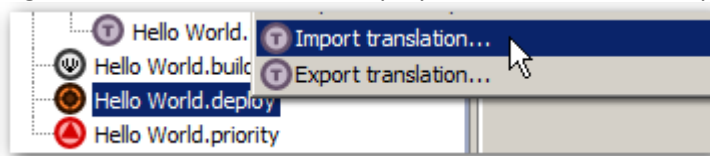


- d. Press *Next* to start the export, then *Finish* to close the export dialog box.

3. Open up the .trans file in a text editor and translate it, all text that is added inside the *TranslatedText*-tag will be included in the text import.

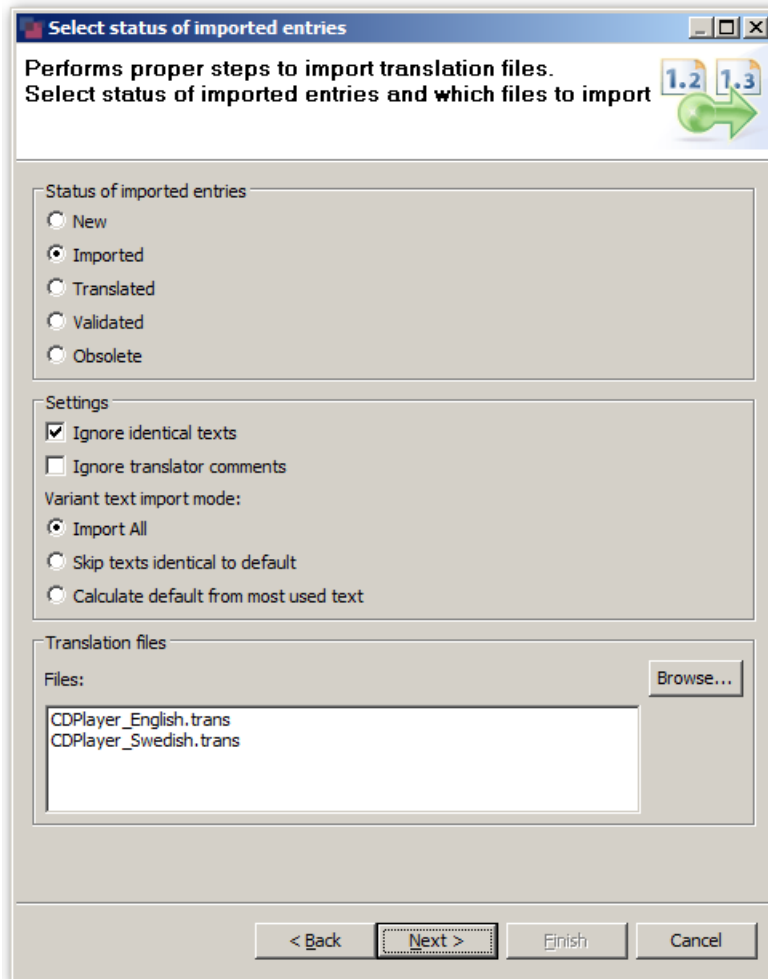
```
<TranslationEntry>
  <TextModuleId>2</TextModuleId>
  <TextModuleName>CDPlayer</TextModuleName>
  <EntryId>1</EntryId>
  <EntryName>MainText</EntryName>
  <VariantId>1</VariantId>
  <VariantName>Default</VariantName>
  <NumberOfLinesAvailable>0</NumberOfLinesAvailable>
  <ParentLanguageTranslation>CD Player</ParentLanguageTranslation>
  <TranslatedText>CD Spelare</TranslatedText>
  <Documentation></Documentation>
</TranslationEntry>
```

4. When satisfied with your work, go back to the Editor and import it back in.
 - a. Right-click the “Hello World” deployment file and select *Import translation...*

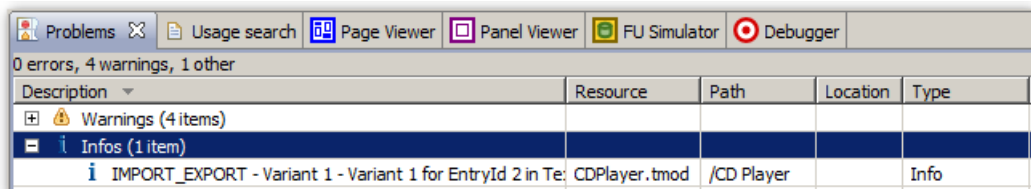


- b. Select all languages and Text Modules and press *Next*.
 - c. Set the status that all imported entries will use to *Imported* and import mode to *Import All*, also check *Ignore identical texts*.

- d. Select the .trans-files that contains the translations.



- e. Start the import process by clicking *Next*. Press *Yes* in the dialog window informing about uploading changes to the project. Then press *Finish* to close the import dialog box.
- f. All changes will be displayed in the *Problems* tab, double-click an entry to review the changes.

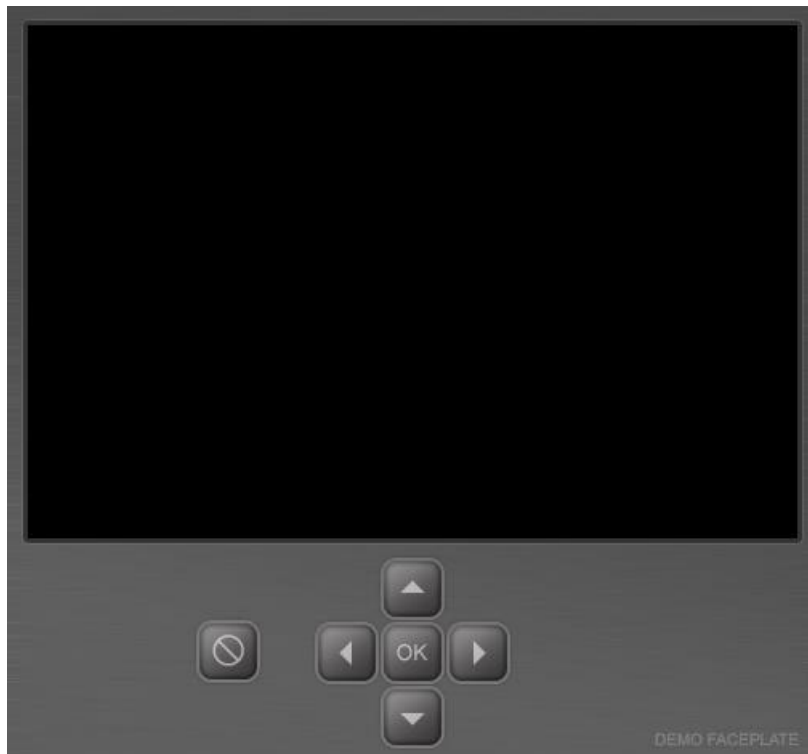


6 FACEPLATE – EXERCISES

6.1 EXERCISE BC6 – CREATE A FACEPLATE

We are going to go back to the CD Player HMI we are creating, and create a faceplate that we can use to simulate physical key presses. (Our HMI is a hybrid one – something that is supported out of the box with Populus.)

To set up a simulation faceplate, we have been provided with a mock-up picture of the real system's front panel. This picture is located in the "CD Player Common" project and is called "faceplate.bmp".



The large black area is obviously the display area, and the buttons represent physical keys.

1. Start by creating a faceplate file, click *File > New > Other...* and select *Faceplate* from the list of objects.
2. Put the faceplate file in the "CD Player" project and just call it "faceplate.faceplate".
3. In the Faceplate Editor, set its image to be the "faceplate.bmp" file within "CD Player Common".
4. Set up the display area to match the black area in the picture. Its dimensions are 480 x 320 but is a bit offset from the top.
5. Next, create the "key info" areas by clicking on the *New Key Info* button. Start with creating the "Up" button. It is important that the Control ID's in the DDH file match the ID's here, and for that reason it is best to put them in the following order (which is the order the following exercises assume): *Up, Right, Down, Left, OK* and last *Exit*. (Control ID 1, 2, 3, 4, 5 and 6 resp.)
 - a. Click *New Key Info*.

- b. Drag and resize the area to match the desired clickable area of the current button.
 - c. Give it a name (for documentation purposes).
 - d. Leave the type set to *Button*. (It is possible to simulate rotary knobs also, by setting up regions that send control events for clockwise and counter-clockwise rotation respectively.)
 - e. Repeat a-d for all keys.
6. Save the file.

The faceplate can be used as-is from within the Editor, by connecting to the Engine using the FU Simulator. Once connected in Manual mode (not running a script), the faceplate becomes active within the Faceplate Editor and the clickable areas send their designated control events to the connected Engine. This is very convenient when connecting to an actual Target environment.

For the Windows Engine, it is possible to have it show the created faceplate as part of its window in Windows. For this to work, the faceplate needs to be serialized.

1. Serialize the faceplate. Click on *Serialize...*
2. This dialog lets you select any folder on your hard drive. To store the resulting `.fbin` file within the "CD Player" project folder find
`C:\PopulusTraining\CD Player.`
3. With the `faceplate.fbin` in place, there are two parameters that need to be updated for the Engine. This is made in the `default.cfg` file, either with a regular text editor or by editing the file from within the Editor itself. Right-click on it and select *Open With -> Text Editor*.

```
# Faceplate binary serialization. (Required on Windows only)
facebin = .\faceplate.fbin
```

```
# Faceplate bitmap file. (Required on Windows only)
faceplate = ..\CD Player Common\faceplate.bmp
```

Hint: If you have a `default.cfg` file which differs from `".\default.cfg"` then choose *Specific configuration file* and select your `default.cfg` file via file browser in the *Preferences > Run* dialogue.

Now run the Engine. The faceplate shows up with the Engine window and if you hover your mouse, the click regions reveal themselves.



Here is a recap of how to get it running (You could also use F9 to re-serialize and run the selected deployment):

- From the *Target* menu, select *Serialize...*
- Correct any remaining verification errors (needs to rerun serialization after correction).
- From the *Target* menu, select *Run Preferences...*
- Change the configuration file to be the `default.cfg` file found in the “CD Player” project (make sure this is the one we modified in this exercise).
- Run the HMI (*Target > Run*).
- Start the CD FU executable (to run the simulated FU) if that is not running. You can find it in
`C:\Program Files\Luxoft Populus xx\Exercises\FU\FUCD.exe`
- Start the Settings FU executable, if it is not running.
`C:\Program Files\Luxoft Populus xx\Exercises\FU\FUSettings.exe`

Note: beginning with release 4.46.0 examples and exercises can be installed to a user-defined directory. The default location is `c:\Users\<user name>\Luxoft Populus xx`.

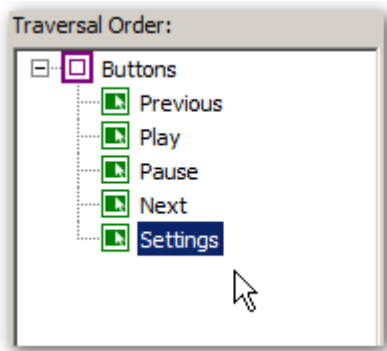
However, nothing happens when regions are clicked. The next exercise will address this...

7 HMI CREATION DETAILED – EXERCISES

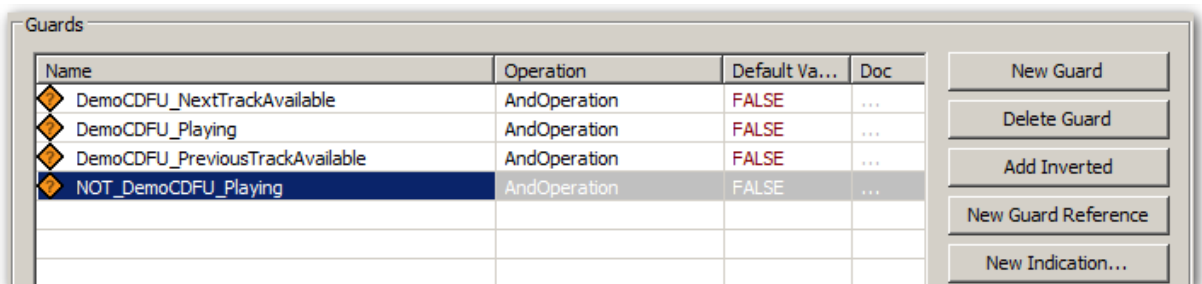
7.1 EXERCISE C7.1 – ADD HMI LOGIC

We are going to add focus traversal order and button enablement to our CD Player HMI. Also, we are going to create the “Settings” profile, which will contain all the settings available in our system. This profile can be reached from the “Settings” button we created.

1. To set up focus traversal for our CD Player page, navigate to the page using *Visual Model* and press *EDIT*.
2. Right-click on “Buttons” panel in the *Z-order* view and select *Add to Traversal Order*.
3. Right-click on each of the individual five fields and do the same to them. They should all be in the order they appear (or the user might get confused).

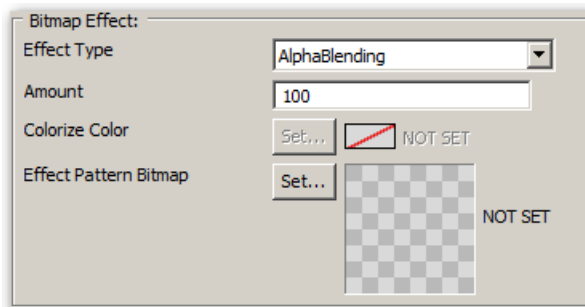


4. To change the runtime enablement of buttons, first we need to set up the Guards that will be used for this.
 - a. Go to the *Guards* tab.
 - b. Press *New Indication* to create Guards for NextTrackAvailable, PreviousTrackAvailable, and Playing (from the DemoCDFU).
5. Select the Playing guard and click on *Add Inverted* to create NOT_Playing.



6. Select each of the buttons and set its corresponding *Enabled* Guard.
 - a. In the *Details* pane for each field, expand *Active*.
 - b. Set the *Enabled* Guard.

- c. Leave the “Settings” button disabled as we set it in a previous exercise. (Enabled Default = false)
7. Go to the *Bitmaps* tab and the icon bitmaps. We are going to make it so that these icons appear dimmed as well, when the button is dimmed out. It is possible to do this without having an actual dimmed out bitmap file ready.
 - a. Open the *Bitmap Definition* by double-clicking in the *Default (Active Skin)* column for each of the five icon bitmaps (PrevIcon, NextIcon etc.).
 - b. Press *Set...* for the disabled state, and make sure that the same bitmap file is set as in the default state (this will be selected automatically).

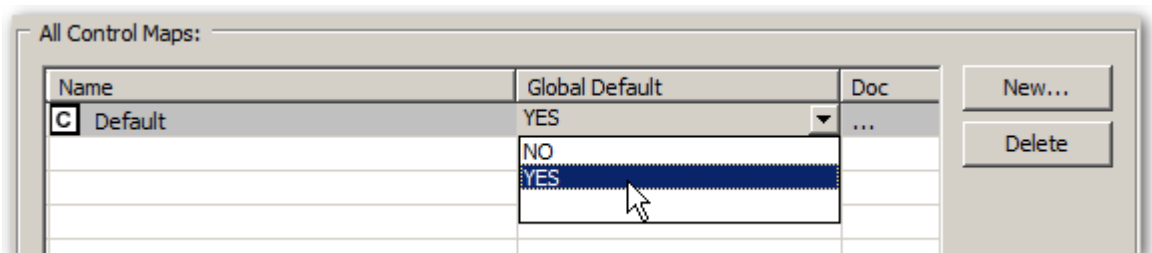


- c. Set up a *Bitmap Effect* which alpha blends the bitmap with value 100 (equals a factor of 100/255).
8. For the user to traverse through the focusable fields, we will set up a Control Map.
 - a. Go to the *Control Maps* tab.
 - b. Add controls for the physical keys available. These must be set up as below to work with the faceplate that we set up in the previous chapter:

Defined Control IDs:

Control ID	Name	Type	Timeout	Doc
1	Up	Button	NOT SET	...
2	Right	Button	NOT SET	...
3	Down	Button	NOT SET	...
4	Left	Button	NOT SET	...
5	OK	Button	NOT SET	...
6	Exit	Button	NOT SET	...

- c. With these in place, create a new Control Map. Make sure it is the global default. Call it “Default”.



- d. Add control map entries for “2-Right” and “4-Left” buttons for Control Event *Pressed* to perform an Internal Action *Focus Next* vs. *Focus Previous*. Leave *Consume* set to NO.
- e. For selection, first add an entry for the “5-OK” button *Pressed* to issue an Internal Action *SetSelected*.

ControlMap entries for Default:

Control ID	Guard	Consume	Control Event	Repeat Rate	Action	Priority
2 - Right	NOT SET	NO	Pressed	NOT SET	[FocusNext] + []	NOT SE
4 - Left	NOT SET	NO	Pressed	NOT SET	[FocusPrevious] + []	NOT SE
5 - OK	NOT SET	NO	Pressed	NOT SET	[SetSelected] + []	NOT SE
5 - OK	NOT SET	NO	Released	NOT SET	[SetUnselected] + []	NOT SE

- f. Next, add another for the *Released* control event that issues an Internal Action *SetUnselected*.
9. Serialize and run in Engine. Notice how it both responds to touch and to the physical faceplate at the same time.

Here is a recap of how to get the HMI running in the Engine (You could also use F9 to re-serialize and run the selected deployment):

- a. From the *Target* menu, select *Serialize...*
- b. Correct any remaining verification errors (needs to rerun serialization after correction).
- c. From the *Target* menu, select *Run Preferences...*
- d. Change the configuration file to be the `default.cfg` file found in the “CD Player” project (make sure this is the one we modified in previous exercise).
- e. Run the HMI (*Target > Run*).
- f. Start the CD FU executable (to run the simulated FU) if that is not running. You can find it in
C:\Program Files\Luxoft Populus xx\Exercises\FU\FUCD.exe
- g. Start the Settings FU executable, if it is not running:
C:\Program Files\Luxoft Populus xx\Exercises\FU\FUSettings.exe

Note: beginning with release 4.46.0 examples and exercises can be installed to a user-defined directory. The default location is `c:\Users\<user name>\Luxoft Populus xx`.

7.2 EXERCISE C7.2 – ADD A SETTINGS PROFILE

1. We will create a profile for all our Settings. By making this a profile, it can be reached in a generic manner from all our “apps”, and returning back from the Settings profile will take the user back to where they were just before entering it.

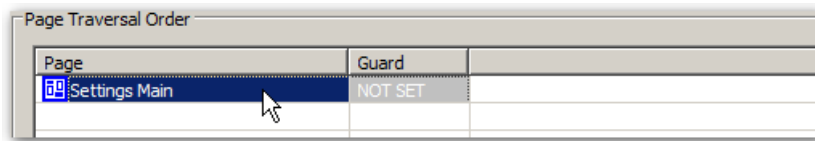
Creating a profile, in turn, calls for a new page. This new page will have a header very similar to the player. To minimize work, we just make a copy of the title panel on the “CD Player” page.

- a. Make a copy of the “CD Title” panel. Select, right-click and choose *Edit > Copy*.
- b. Go to the *Pages* view in the Visual Model (leave the *Editor* mode).
- c. Create a new Page “Settings Main”.
- d. Edit it and set up its background bitmap (same as “CD Player”).
- e. Right-click in the editor area and select *Edit > Paste*. This will paste a copy of the “CD Title” panel, and call it “Settings Title”.
- f. Change the text for “CD PLAYER” to “SETTINGS” by changing the corresponding StaticTextField Text ID to Text Module “Settings” and its title entry.

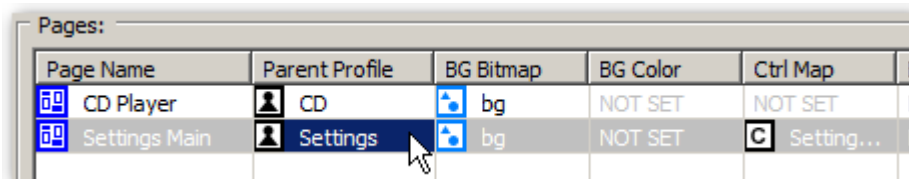


- g. This will be the main page for our new Profile. Which means it is time to create the profile: Go to the *Profiles* tab and click on *New Profile*.
- h. Call it “Settings”.
- i. Set its parent Frame to “Default”. This means the one frame now contains two profiles, of which only one can be active at a time.

- j. Scroll down to *Page Traversal Order* and add the “Settings Main” page as the first page for this profile.

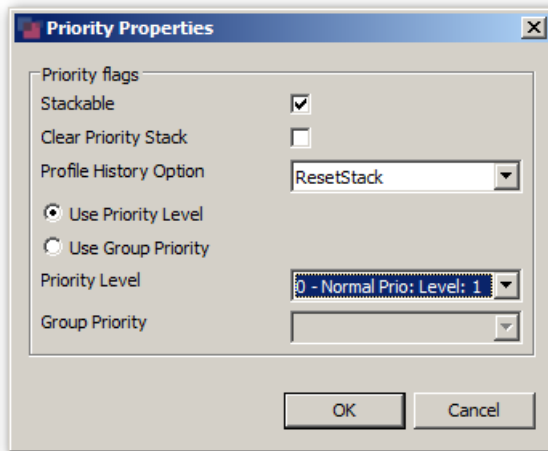


- k. Go to the *Pages/Panels* tab and set the *Parent Profile* of the “Settings Main” page to the “Settings” profile.

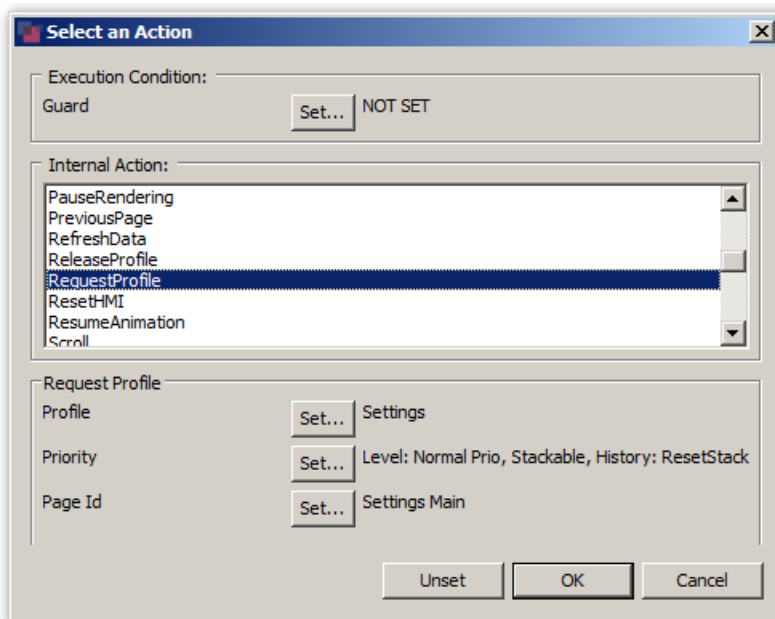


2. To be able to change profile from the HMI, we need an internal event.
Create a new FU Interface for an InternalFU.
 - a. Create a new sub folder “FU” within the project “CD Player”.
 - b. Next, right-click on the “FU” folder and select *New > Other...*
 - c. Select *Functional Unit Interface* and give it the name “Internal.fuclass”. FU ID must be 255.
 - d. Add the new FU Interface to the current Deployment (right-click and select *Add to active Deployment*).
 - e. Add an Event called “Request Settings”.
 - f. Add another Event called “Release Settings”.
 - g. Save and close.
3. Now map these new Events to profile request and release messages in the *Events* tab.
 - a. Go to the *Events* tab.
 - b. Click on *Add...* in *Global Events* group.
 - c. In the dialog that opens choose the “Request Settings” event in “InternalFU”.
 - d. Click *OK*.
 - e. Double-click in the *Action* column.
 - f. In the dialog that opens click *Add...* in the *Internal Actions* section.
 - g. Select “RequestProfile” and click the *Set...* button in the *Profile* line.
 - h. Choose “Settings” and click *OK*.
 - i. Click the *Set...* button in the *Priority* line.

- j. Select *Priority Level* “0 – Normal Prio: Level: 1”. Click *OK*.

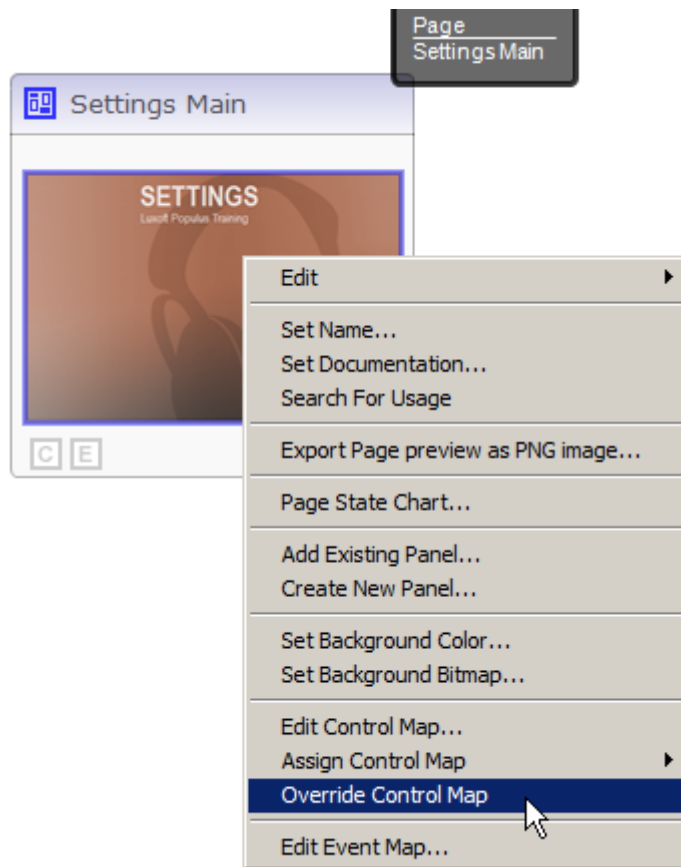


- k. Click the *Set...* button in the *Page Id* line.
l. Select “Settings Main” and click *OK*.



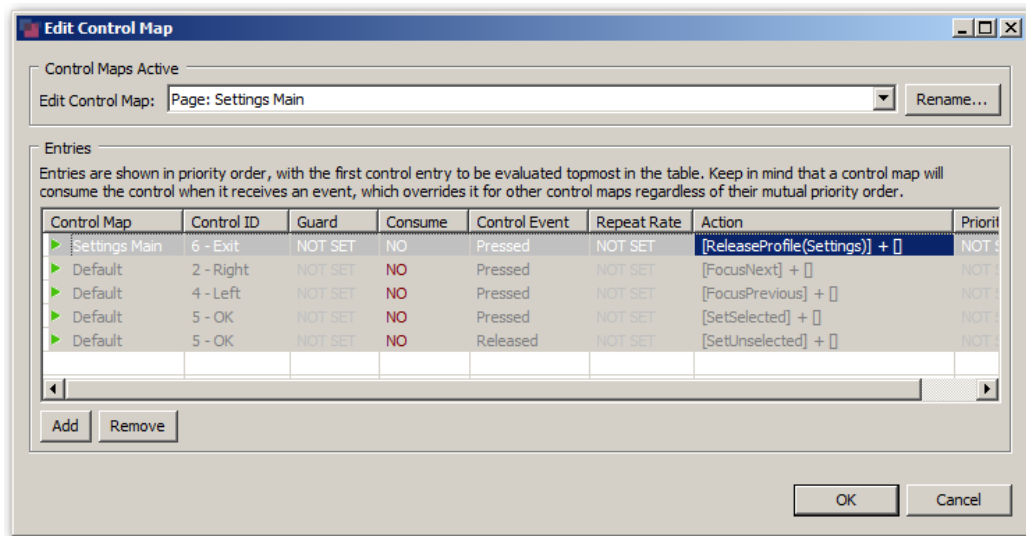
- m. Click *OK* two times to close the *Select and Action* and *Set Actions* dialogs.
n. Repeat steps b-m for “Release Settings” but use “Release Profile” action instead (note that this time you won’t need to set up *Priority* and *Page Id*).
4. Assign a select action that makes a Settings profile request when pressing the “Settings” button.
a. Go back to *Visual Model* tab and select the “CD” profile and “CD Player” page. Edit it.
b. Select the “Settings” button field and scroll down to its *Active Control Field Basic Properties* in Details.
c. Add an Internal Action *InternalEvent* to invoke the “Request Settings” event we created in Internal FU.

- d. It is time to make the “Settings” button enabled, so set its *Enabled Default* to *Yes*.
5. Try navigating to (using right/left) and selecting (using OK) the “Settings” button, and verify that the “Settings” profile is activated.
 - We currently have no way of navigating out of the Settings profile!
6. To be able to exit from the “Settings” page, the “Exit” button should be mapped to perform a “Release Settings” – but this should only be applied to the “Settings Main” page.
 - a. Override the *Control Map* for the *Page* “Settings Main”.

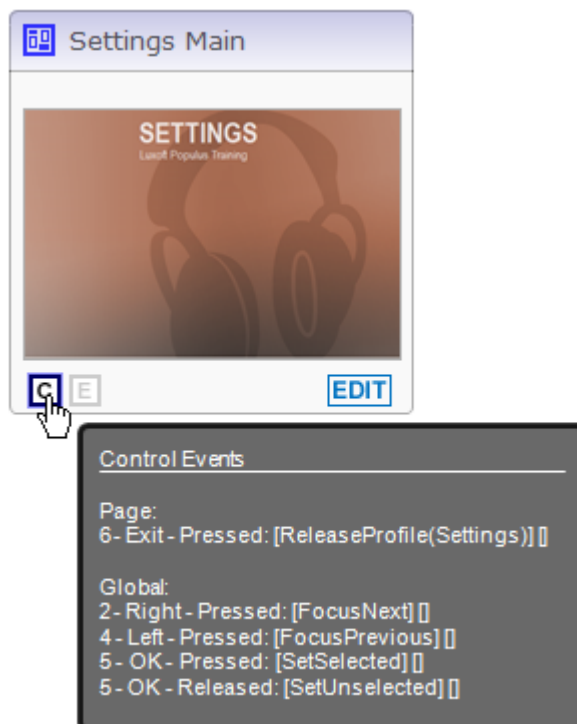


- b. In the dialog window *Add Control Map* that opens, enter the name for a new *Control Map*. Call it “Settings Main” as this is the control map for the page.
- c. Right-click the *Page* again and select *Edit Control Map*.

- d. Add *Internal Event* (“ReleaseProfile”) as the *Action* for *Pressed* on Control “Exit”.



7. Click OK to close the *Edit Control Map* dialog and check effective *Control Map* for the “Settings Main” page by hovering the mouse over the [C] box. Compare to the “CD Player” page, which only falls back to the default map.



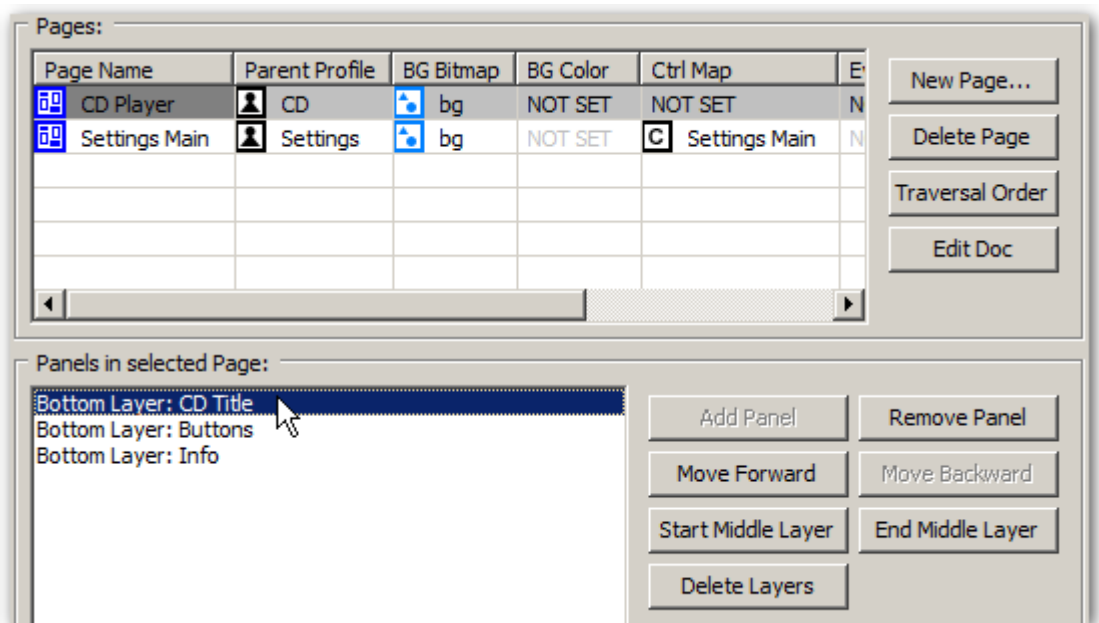
8. *Reserialize and Run*. Now it is possible to come back to the “CD Player” page from the “Settings” profile, by releasing the priority request for the “Settings” profile.

7.3 EXERCISE C7.3 – ADD PROFILE TRANSITION EFFECT

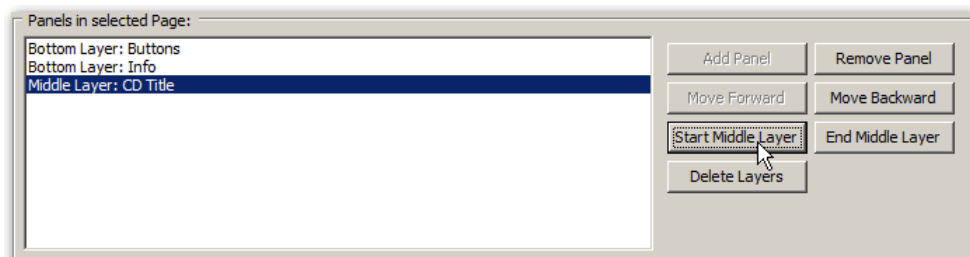
We are going to enhance the user experience in our CD Player HMI by adding a transition effect to the profile change (we have two profiles to transition between).

To be able to do more complex transition effects it is best to partition the page into layers that can be animated independently.

1. Setup layers for “CD Player” page so that the “Button” Panel and “Info” Panel are part of the bottom layer and the “CD Title” is in the middle layer.
 - a. Go to the *Pages/Panels* tab.
 - b. Select the “CD Player” page in the upper table.



- c. Select the “CD Title” panel and move it forward until it is last in the list. Now click on *Start Middle Layer*, so that the “CD Title” panel is alone in the middle layer of the page.



2. Setup layers for the “Settings Main” page so that the title panel is in the middle layer as well.
3. Go to the *Visual Effects* tab and set up a *Profile Transition Effect* to use between the profiles.
 - a. Click *New Transition*.
 - b. Set *From Profile* to “CD” and *To Profile* to “Settings”.
 - c. Design the visual effect by double-clicking in the *Transition* column.

-
- d. Create an effect by fading/moving the old Bottom layer while fading/moving in the new Bottom layer. Let the middle layer slide between old and new page.

(In this case there is no content in the “Settings” page Bottom layer).
 - e. Preview the effect and press *OK* when satisfied.
 - f. Make a copy of it, and set its *From Profile* to “Settings” and *To Profile* to “CD”.
 - g. Now design an “opposite” transition effect (you may use the *Reverse* function).
4. Serialize (and correct any remaining verification errors!) and run in Engine.
 5. Go to “Settings” by selecting the “Settings” button, and watch your effect.

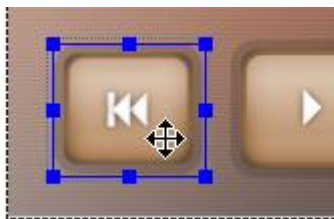
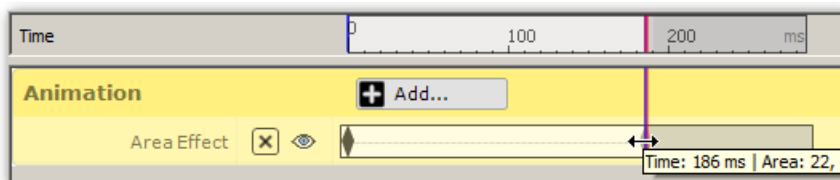
7.4 EXERCISE C7.4 – ADD BUTTON ANIMATIONS

Now that we have learned about Action Triggers and how to start animations, we can apply this to the CD Player HMI. We are going to add two animations; one for when the button is pressed and one for its release.

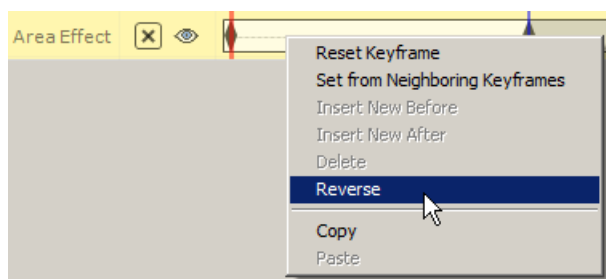
1. Start off with defining the animations. Go to the *Visual Effects* tab and select the *Field and panel animations* sub-tab.
 - a. Create a new Animation ID, call it “Button press.”
 - b. Assign it to one of the buttons in the “Buttons” panel, which will act as a template.

Name	Panel	Field	Scene	End State	Use Absol...	Visual Effect	Doc
Button press	Buttons	Previous	NOT SET	KeepEndState	NO	SET	...

- c. Design the button press effect to use an *Area Effect* that will shrink the button a bit towards the end. Set it to *Keep End State*.



- d. Create another Animation for “Button release” by making a copy of “Button press”. (Rename the copy “Button press 2” and use *End State GoToNormalState*.)
 - e. Edit this animation to do the opposite (expand up again), by right-clicking in the *Area Effect* and selecting *Reverse*.

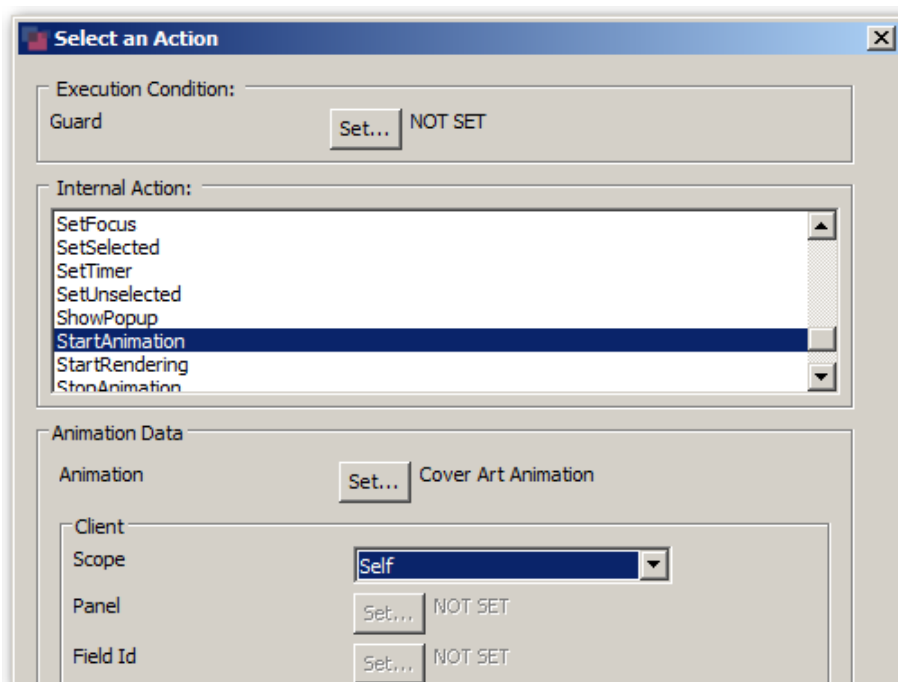


2. Go to the *Visual Model* and Edit the “CD Player” page.
3. Set up Action Triggers on each of the five buttons, to start “Button press” OnSelectGained and “Button release” OnSelectLost.

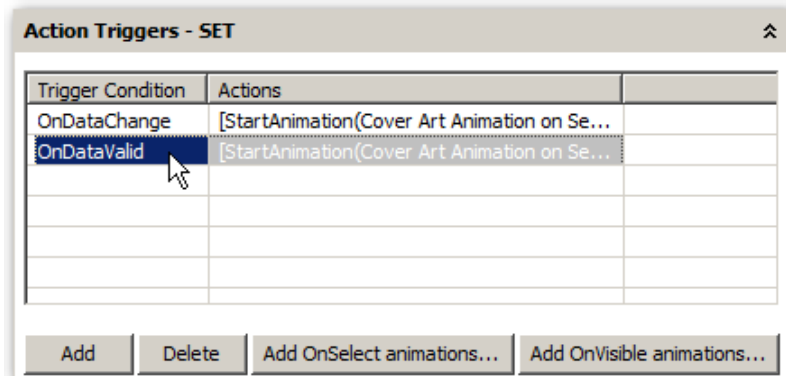
- a. Select the field, and go to the top of the *Details* pane and expand *Action Triggers*.
 - b. Use the *Add OnSelect animations* shortcut button.
 - c. Select the corresponding animations. Make sure the animation client is set to *Self*, so that these action triggers can be copy/pasted.
 - d. Next, make a copy of the two action triggers by selecting them both and choosing *Copy* (Ctrl-C).
 - e. Go through each of the four remaining fields and paste the action triggers in their *Action Triggers* table.
4. Serialize and correct any remaining verification errors. Run and try out the button animations by clicking them with your mouse and also by using the faceplate buttons.

OPTIONAL EXERCISE:

5. The information panel should only be showing when there is a track playing. Set up its *Visibility Guard* to “Playing” and define *OnVisible* animations (perhaps fade in/fade out?) for it. Remember to use the panel’s *Action Trigger*.
6. It’s also possible to animate data driven fields when their data changes. Create an animation that bounces the “Cover Art” field in the “Info” panel when its data has changed.
 - a. First, add and design the animation in the *Visual Effects* tab. Go to *Field and panel animations* and click *New Animation*. Set the “Cover Art” field as the template, and then design the bounce as an Area Effect.
 - b. Add an action trigger for the condition *onDataChange* in the “Cover Art” field, by clicking on *Add* and then selecting that trigger condition.
 - c. Set up an Internal Action *Start Animation* with the created animation and on *Self*.



- d. Make a copy of this action trigger, and paste it.
- e. Change the duplicate action trigger's condition to *onDataValid*, so that we get the bounce also when there was no cover art on the track before.



7. *Target > Serialize and Run.*

8 RUNTIME

8.1 EXERCISE BC8 – SIMULATE A FUNCTIONAL UNIT

We are going to simulate the DemoCDFU in this exercise, instead of running the executable. What we want is to have only the Engine running the HMI and not the FUs. In order to accomplish that, we need to close down the FU CD executable.

1. Close the CDFU console window, effectively exiting that FU.
2. Back in the Editor, always make sure that you have the same Deployment active as the one serialized and run in the Engine. In our case, this should be no problem.
Go to the FU Simulator tab and set up the connection to 127.0.0.1 (localhost) and port 29101.
3. Make sure the Engine is running (*Target > Run*).
4. Click on *Connect*.
5. Next, click on *Select FUs...* to select the FUs that should be simulated. This makes it possible to combine simulated FUs with real ones.
6. Choose to simulate only the “DemoCDFU”.
7. Click on *Manual*.
8. Open up the DDH Runtime file that we created in Exercise A4.2. Make sure it has been set as the current, by right-clicking it and choosing *Set as Active Runtime Data File*.
9. Select the CD FU (192). Try modifying the *Playing* indication. Notice how the HMI changes. Change the Data that we use in the HMI as well and notice the changes.
10. Try the Debugger as well. Setup the connection (same IP and port as for the FU Simulator) and press *Connect*.
 - a. Get the status of all Guards by clicking on *Get Guard*.
 - b. Get all Data by clicking on *Get Data*.
 - c. Etc...

9 [DEPRECATED]

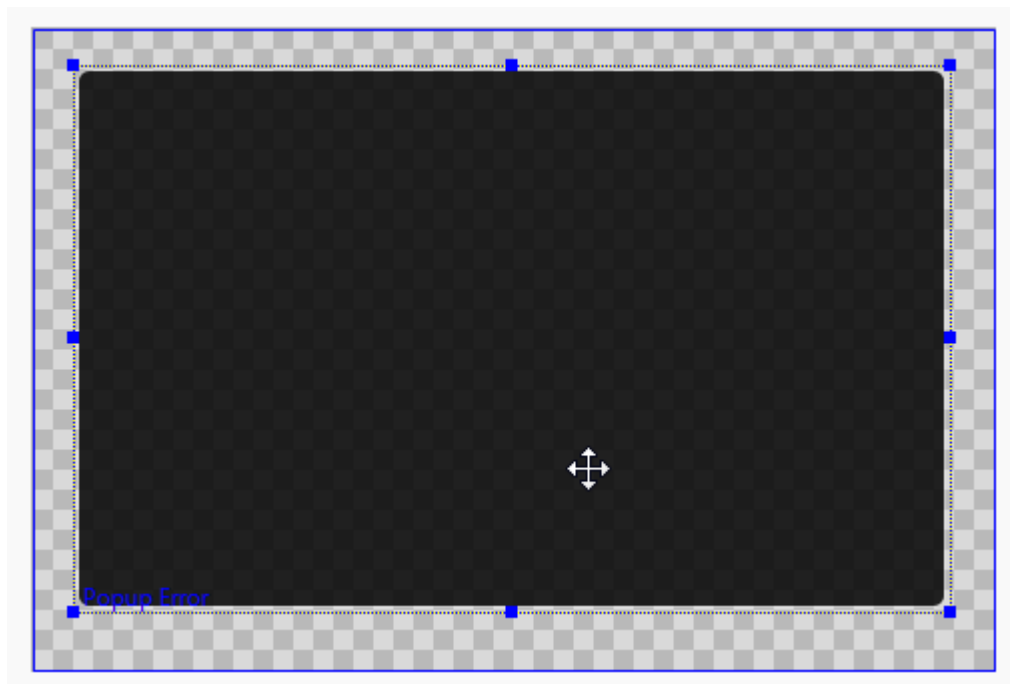
10 HMI CREATION ADVANCED

10.1 EXERCISE C10.1 – ADD A POPUP

We would like to show an error message in the event that there is a read error in the CD Player. First design the popup panel for such an error message, then set up the popup definition. Should this error be possible to redisplay, for instance?

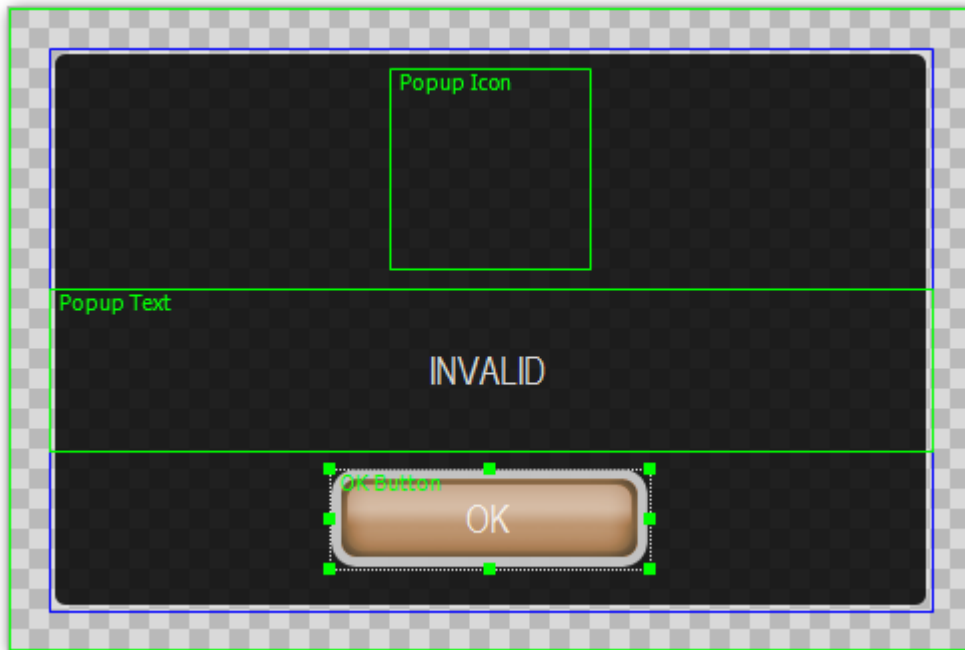
Utilize popup panel reuse, so that the popup defines the Text ID for the actual message. Next, map the error event from the DemoCDFU to show this popup.

1. In *Visual Model*, navigate to *Panels* and right-click and choose *New Panel*. Call the new popup “Popup Error”.
2. Edit the new panel (we will use this as the popup panel).
3. Go to *Bitmaps* tab and add the file “HeavySemiTransparentbackgro.png”. Set it to *Stretch With Border* and a border width of 10 pixels.
4. Add the “ErrorIcon.png” as well.
5. Back in the *Visual Model* Editor, set the panel background to the “HeavySemiTransparentBackgro” bitmap.
6. Change the size of the panel so it doesn’t entirely cover the screen.



7. Add a new *DynamicBitmapField* for the icon called “Popup Icon”. Set its data to FU 0 and “PopupBitmapID”. Make it 100 x 100 pixels in size.
 - a. Set *Use Default Data* for *Rotation Angle* field. Select value type *Integer* and fill in the value of 0.
8. Add a new *DynamicTextField* below it and assign it FU 0 and “PopupTextID”.

- a. Call it “Popup Text”.
 - b. Set its horizontal alignment to *Center*.
9. Add an *Active Control Field* at the bottom that will acknowledge the popup (set its *Select Action* to perform an Internal Action *AcknowledgePopup*). Name it “OK Button”.

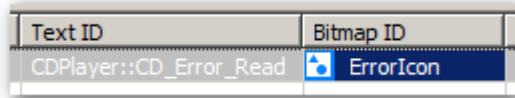


- a. Set its *Outline Bitmap* to the button bitmap we used for the CD Player buttons
 - b. In *Text Properties*, check *Enable Text*, and select the “OK” text entry in the “CDPlayer” text module.
 - c. Set the text to be centered by changing the *Horizontal Alignment* to “CenterAlignment”.
10. Add the “OK Button” field to the panel’s field traversal order, so that it will receive initial focus.
 - a. Right-click and select *Add to Traversal Order*.
11. This popup will now act as a template for all error message popups, containing an icon and a message text.

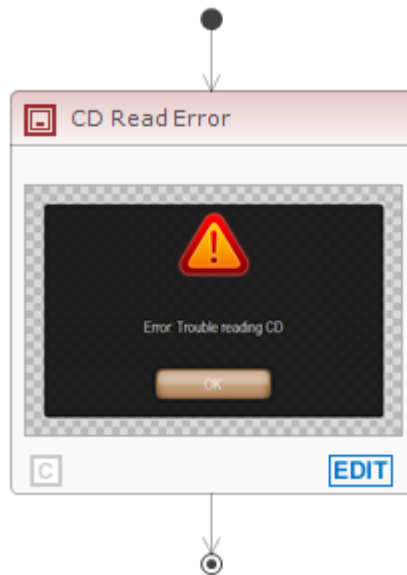
Go to the *Popups* tab and create a new Popup.

- a. Call it “CD Read Error”.
 - b. Assign it the popup panel we created.
 - c. Leave the *Popup Title Text ID* as *NOT SET* (we don’t use that in our template).
 - d. Set up *Popup Text ID* to point to the error message in “CD Player” text module.

- e. Set up Popup *Bitmap ID* to point to the Bitmap ID with the error icon.



- f. Set up *Redisplay* to “Yes” and *Priority* to Normal.
- g. Feel free to setup any other properties, like *Timeout* etc.



12. Map the error event to show this popup.
 - a. Go to the *Events* tab.
 - b. Click *Add...* and select DemoCDFU event “CDReadError”.
 - c. Add Internal Action “Show Popup” and select the popup created.
13. Add transition effects for popups
 - a. Go to the *Visual Effects* tab and *Change Popup Transition* subtab.
 - b. Create a new transition effect for Show Popup; maybe bounce up by using an Area Effect...
 - c. Create a new transition effect for Hide Popup, maybe just fade it out.
14. Done, serialize and run. The implementation of the CD Player is set up to simulate a read error whenever track 3 is played.

(Make sure the FU Simulator is not connecting to the Engine before starting the FUCD again.)

Here is a recap of how to get the HMI running in the Engine (You could also use F9 to re-serialize and run the selected deployment):

- a. From the *Target* menu, select *Serialize...*
- b. Correct any remaining verification errors (needs to rerun serialization after correction).

-
- c. From the *Target* menu, select *Run Preferences...*
 - d. Change the configuration file to be the `default.cfg` file found in the “CD Player” project (make sure this is the one we modified in this exercise).
 - e. Run the HMI (*Target > Run*).
 - f. Start the CD FU executable (to run the simulated FU) if that is not running. You can find it in
`C:\Program Files\Luxoft Populus xx\Exercises\FU\FUCD.exe`
 - g. Start the Settings FU executable, if it is not running.
`C:\Program Files\Luxoft Populus
xx\Exercises\FU\FUSettings.exe`

Note: beginning with release 4.46.0 examples and exercises can be installed to a user-defined directory. The default location is `c:\Users\<user name>\Luxoft Populus xx`.

10.2 EXERCISE C10.2 – ADD A SETTINGS MENU

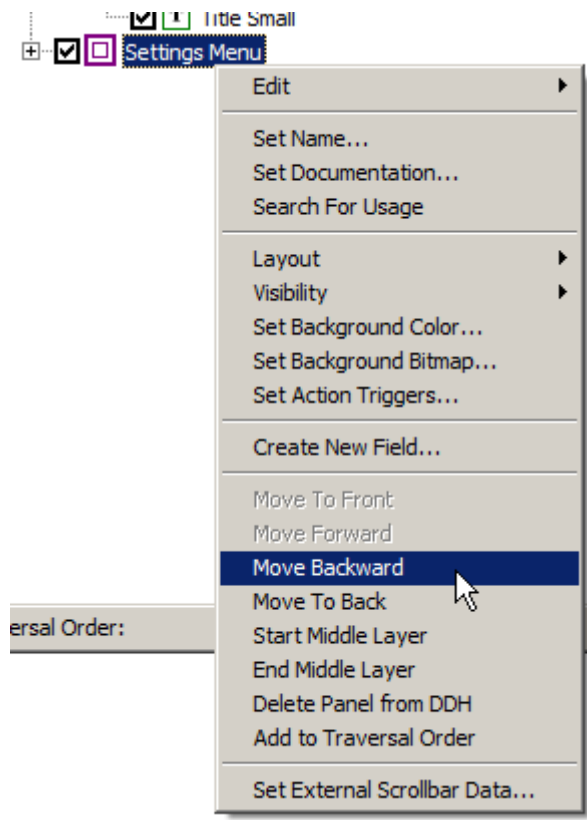
Our Settings page is going to be extended with the possibility to actually perform some settings. We are going to add a menu to the Settings page which will contain the possibility to set current language, set the time, set units and go back to the CD Player page. This is the end result we have in mind:



(The option to set current language is actually the only option we will implement at this stage. This item will open up a “sub menu” by making a page change to a new page.)

1. Go to the *Visual Model* tab and navigate to the “Settings Main” page via *Pages*, and click *EDIT*.
2. Create a new panel for the menu called “Settings Menu”

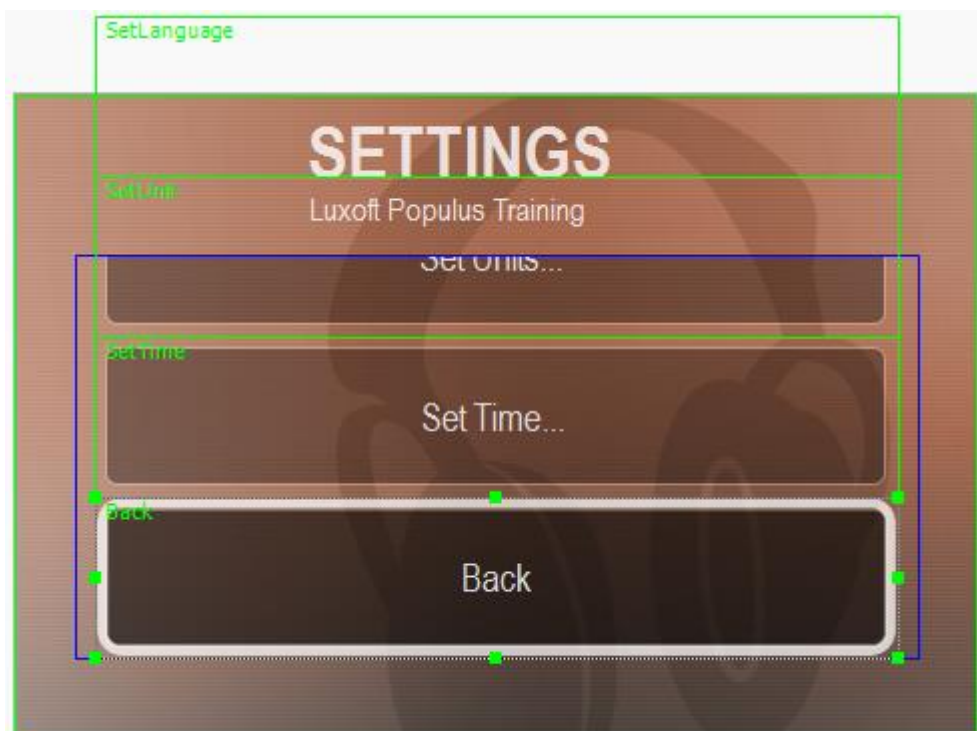
3. Change the Z-order so that it appears behind “Settings Title” by right-clicking the new panel in the Z-order list and choosing *Move Backward* or *Move to Back*.



4. Adjust the size of the panel so that it is centered under the title.



5. Go to the Bitmaps tab and add a new bitmap to decorate the menu: click Add and add the image "menuitem_normal.png", then rename it to "Menu Item". This bitmap will use 3 states; Default, Focused and Selected:
 - a. Set "menuitem_normal.png" as *Default Bitmap*: use *Stretch With Border*, with 15 pixels border;
 - b. Use "panelmenuitem_focused.png" and "menuitem_active.png" for the focused/selected state, using same settings as for the default state.
6. Back in the "Settings Menu" panel: set the panel's *Layout Type* to *Row*, and enable the *Row Layout Configuration* further down:
 - a. Set *Row Type* to *Vertical*;
 - b. Set *Alignment* to *Center*;
 - c. Leave *Spacing* at 0 pixels.
7. Add the first menu item as an *ActiveControlField*. Call it "SetLanguage".
8. This new field becomes too large vertically, so adjust the size to be around 40% of the available viewport (panel height).
9. Set the field's *Outline Bitmap* to "Menu Item" prepared a few steps back, and enable it to show text from the Text ID for "Set Language..." in the Settings text module.
10. For the three remaining items, copy and paste the SetLanguage item we just created.
 - a. Call them "SetUnit", "SetTime" and "Back".
 - b. Assign each of them a corresponding Text ID from the Settings text module.



The Editor preview will reveal the field you select from the Z-order view, by scrolling the row layout.

11. Add the Settings Menu to the Page's panel traversal order, and add each field in turn into traversal order, to make it possible to navigate the menu using the faceplate.
12. For the "Back" item, set its *Select Action* to issue an Internal Action "ReleaseProfile" with the profile "Settings", having the same functionality as the "Back" key earlier.
13. In the "Settings Menu" panel, add an animated focus bitmap:
 - a. Enable *Focus Object Background* in the *Layout Focus Objects* section of the details;
 - b. Set *Focus Object Bitmap* to the "menu_focus_background" bitmap (you'll have to add it separately on the *Bitmaps* tab);
 - c. Set *Animate Focus Change* to YES and select a set a PIDRegulator as *Filter*.

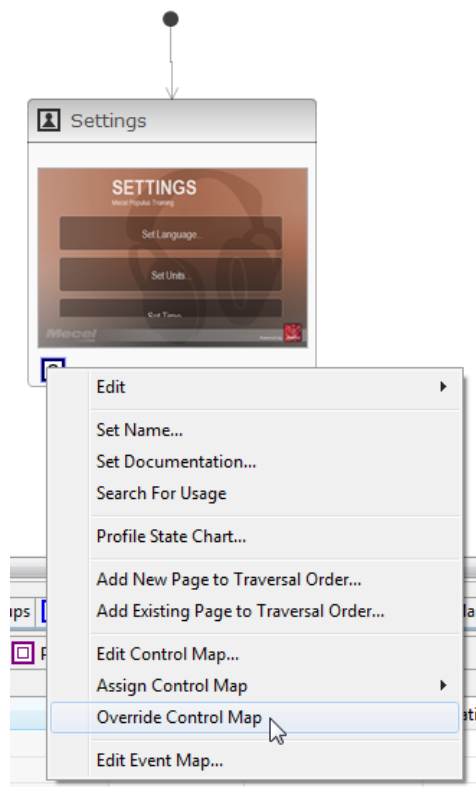
This accomplishes a slide between the fields, as the user traverses between items.


14. Reserialize and Run this HMI. Use the faceplate to navigate the menu. This feels awkward – we haven't mapped the Up and Down keys yet.

For things to be more intuitive to the user, we want to modify the control map so that Up and Down keys can be used to navigate in the menu:

- a. Navigate to *Profiles* and "Settings" profile and override its control map.

(Remember – we made an overriding control map for the page, to be able to press a key to exit the Settings. Now, we want this behavior to work throughout the whole Settings profile.)



-
- b. Create a new Control Map for this profile by pressing *Override Control Map*, call it "Settings".
 - c. Next, edit the control map (Click on the  and *Add* control map "Settings" in the opened dialog). Add *Focus Next* on Pressed for Control "Down". It does not have to have *Consume* set to YES to consume it within this Control Map, so we may leave it at NO.
 - d. Add *Focus Previous* on Pressed for Control "Up".
 - e. Press *OK* to set.
15. Now try this out in the Engine, *Target > Serialize & Run*.

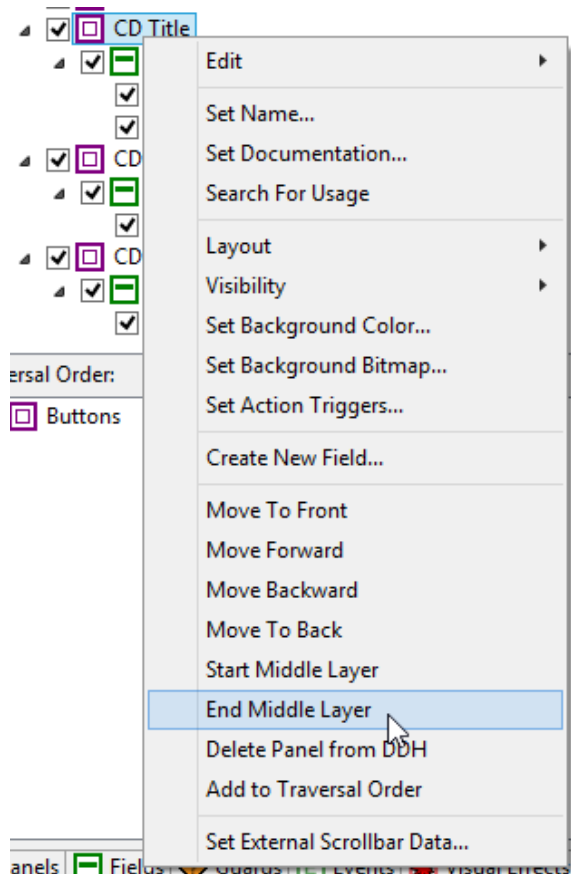
10.3 EXERCISE C10.3 – ADD ANIMATION FIELD

We are going to add an animation field that shows a spinning CD. It should only spin when the CD is playing. This field will be placed to the left of the title in all pages.

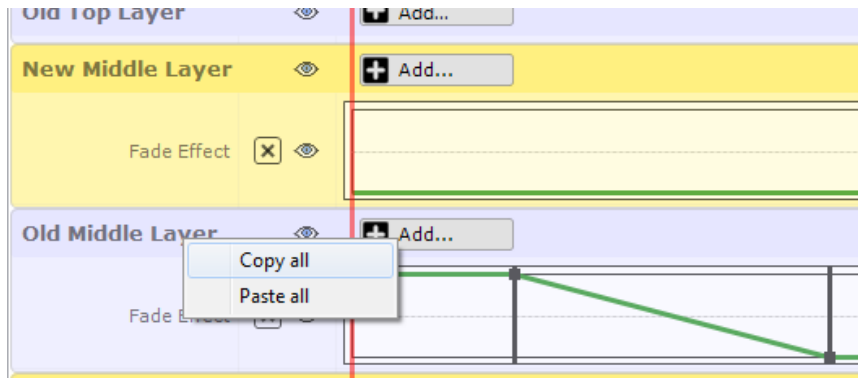
1. Go to the “CD Player” page and create a panel “CD Icon”.



2. Create a new *AnimationField* called “Icon Animation” inside it.
3. Go to the *Bitmaps* tab and add the frames that will be used as the animation. Click on *Add...* and select all the bitmap files called “cdicon***.png”.
4. Back in the *Animation Field* of the *Animation Data* section in details add the bitmap IDs to its sequence.
5. Set it to loop back to frame index 1.
6. Set its Guard to “DemoCDFU_Playing” so that it will spin only when we are playing music.
7. Adjust the panel’s Z-order (*Move to Front*) and make sure that it is part of the Top Layer on the page (select *End Middle Layer* on the “CD Title” panel).



8. Go to the “Settings” page and *Add Existing Panel*. Add “CD Icon”.
9. Make it part of the top layer for these pages as well just as in step 7 above.
10. *Target > Serialize and Run*.
11. In the profile transition effects that we have made, we have no effect applied to the Top Layers. Since we want to have the same fade effect that the Middle Layer uses, copy the old and new Middle Layer effects to the old and new Top Layer by right-clicking in the left side of the animation table and selecting corresponding commands in the context menu.



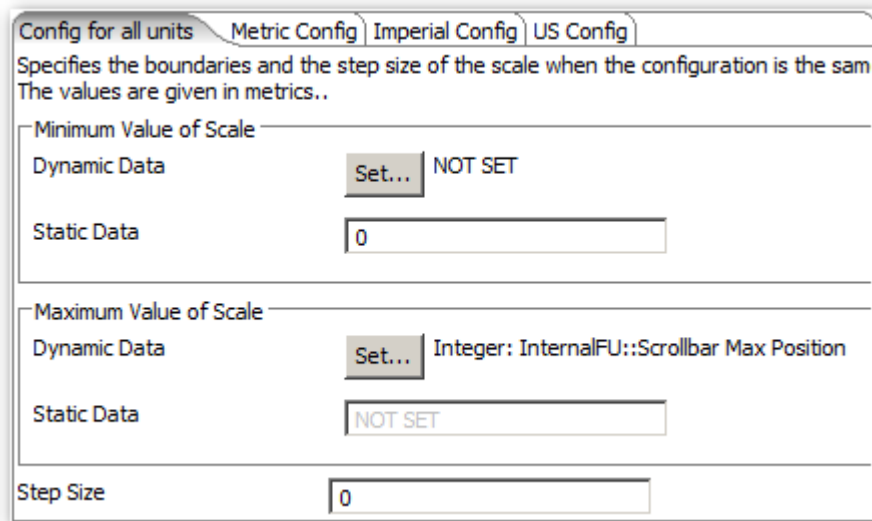
10.4 EXERCISE C10.4 – ADD A SCROLLBAR

In this exercise, we will add a scrollbar to the “Settings Menu”. This is accomplished by using a *GraphicalScaleField* with *Fillbar* appearance that shows data from the *Row Layout* set on the panel. The panel’s row layout can be set up to publish data that the scrollbar can use to present to the user where the viewport is within the entire canvas.

1. To make it possible to publish data, there needs to be data available in the internal FU (255). Open up the “Internal.fuclass” file in the FU folder (inside the CD Player project).
2. In the *DynamicData* tab, add the following Data ID’s:
 - a. “Scrollbar Start Position” as *Integer*;
 - b. “Scrollbar End Position” as *Integer*;
 - c. “Scrollbar Max Position” as *Integer*.
3. Go to the “Settings Main” page through *Visual Model*, and click *EDIT* to edit it.
4. Select the “Settings Menu” panel and set up *External Scrollbar Data* in the *Panel* section of the details:



- a. Choose to set up the Vertical Pixel data.
(Always use pixel data when creating a scrollbar for a view, so that it can reflect the size of the view port compared to the size of the canvas properly.)
 - b. Set *Total Size Data Id* to “Scrollbar Max Position” in internal FU (255);
 - c. Set *Start Pixel* to “Scrollbar Start Position”;
 - d. Set *End Pixel* to “Scrollbar End Position”.
5. Create a new panel for the scrollbar and its field.
 - a. Right-click in the page and *Create New Panel*. Call it “Scrollbar”.
 - b. Position the new panel on the right hand side of the “Settings Menu” panel.
 - c. Set the panel’s *Background Bitmap* to be the “HeavySemitransparentBackgro” bitmap we have used before.
 - d. In this new panel, right-click and select *Create New Field*. Create a *GraphicalScaleField*, and call it “Scrollbar” as well.
 - e. Position the new field within the gray area of the background bitmap.
 6. Set the “Scrollbar” field’s details as follows:
 - a. Under *Scale Data*, set the *Dynamic End Scale Data* to “Scrollbar End Position” in internal FU, and set the *Dynamic Start Scale Data* to “Scrollbar Start Position”.



Config for all units | Metric Config | Imperial Config | US Config

Specifies the boundaries and the step size of the scale when the configuration is the same. The values are given in metrics..

Minimum Value of Scale

Dynamic Data: Set... NOT SET

Static Data: 0

Maximum Value of Scale

Dynamic Data: Set... Integer: InternalFU::Scrollbar Max Position

Static Data: NOT SET

Step Size: 0

- b. Set the *Maximum Value of Scale* to use the Dynamic Data “Scrollbar Max Position” and clear the corresponding *Static Data* field.
7. Now, create an Appearance for the *Graphical Scale Field*, and pick the *Fill Bar* appearance.
8. To show the position of the scrollbar, we first need to add a bitmap for it. Go to the *Bitmaps* tab and add the file “scrollbar_pos_fg.png”.
 - a. Set *Render Type* to *StretchWithBorder*, 3 pixels by entering the settings for the newly added bitmap in the section *Default Bitmap*.
9. Back in the *Appearance* details for the scrollbar field, set up the Appearance as follows:
 - a. Set *Scale Type* to *Vertical Scale* since we want it to indicate positions vertically.
 - b. Check the *Stretch Fill Outline to Range* option and set the *Fill Outline Bitmap* to the “scrollbar_pos_fg” bitmap just created.
 - c. Set *Fill Outline Minimum Size* to 10 pixels so that it will never show up smaller than this.
 - d. Make sure the *Scale Maximum Pixel* corresponds to the height of the field. Approximately around 170 pixels.
10. Move the “Scrollbar” Panel to the back in Z-order by right-clicking on it, and selecting *Move to Back*.
11. With the scrollbar in place, serialize and run the HMI. *Target > Serialize and Run*. Try scrolling up and down with the faceplate keys.

Note: Implemented this way, the scrollbar is only able to show the current scrolling level. To enable scrolling by dragging the scrollbar, a more complicated solution is required.

10.5 EXERCISE C10.5 – ADD A TRACK LIST

We are going to add the possibility to show a list of all tracks in the CD Player. For this view, we are going to create a new page with the track list in it.

There needs to be a way to get to the track list view, which is why we will add yet another button to the main CD Player view. This new button will be positioned above the “Info” panel.

1. Go to the *Visual Model* tab and the *Pages* view. Add a new Page and call it “CD Track List”.
2. Click EDIT on the new page.
3. Set up the same background as we have used for the other pages (“bg”).
4. Set Parent Profile to “CD”.
5. Right-click and add the existing “CD Icon” and “CD Title” panels.
6. Now create a new panel for the track list. Call it “CD Track List”.
7. Create a new *PanelListField* within this panel. Call it “Track List”. Make the area so that it covers most of the screen.



The new panel list shows up red because there are no panels defined for it. The list uses panels to define how each item looks.

8. Create yet another panel and call it “Track Item”. Position it inside the list, and make it the size you would like each track item to occupy within the list. We will make room for 4-5 tracks in the list, so set the panel’s height to about 60 pixels.
(The page needs to be selected to be able to click *Create New Panel*).

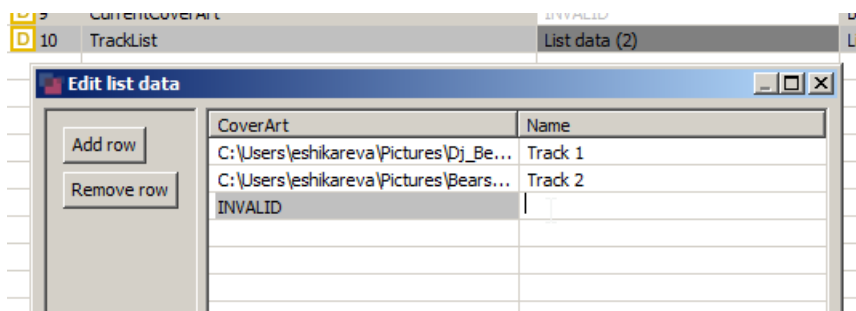
9. Assign the track item panel the “Menu Item” bitmap we used for the “Settings Menu”.



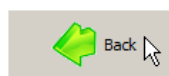
10. Select the “Track List” field again and set up the following in its *Details* view:
- Set *Dynamic Data* to the “TrackList” in the DemoCDFU.
 - Scroll down a bit and find the *Item Panel Set*. Assign the *Default List Item Panel* to the “Track Item” panel we just created. We have now defined all items to use this panel as a template.
 - Remove the “Track Item” panel from the page, since we don’t want to include the panel as a regular page panel.



11. To be able to preview a list in the Editor, there needs to be data set for it in the current DDH Runtime file. This is because the appearance of a list is very much dependent on the data inside it.
- Go to the DDH Runtime file and fill in data for the list.
 - Edit the list data in the table and click on *Add row* inside the dialog to add items.



- c. You may add your own images as cover art, but at least type in a string for each item. E.g. "Track 1", "Track 2" etc.
12. Go back to the DDH file and have a look at your list now. It probably seems to show only one panel, but this is because it is set to be horizontal by default.
 - a. Go to the *Row Layout Configuration* details for the list and change the *Row Type* to Vertical.
 - b. Set *Alignment* to Center.
 - c. Check *Focus Object Foreground* in *Layout Focus Objects* and set the bitmap to the same we used for the Settings Menu, that is "menu_focused_background". Also make sure that *Animate Focus Change* is set to YES by default.
 - d. Select an easing curve of your choosing.
13. Still, we are not seeing any data from our list – which is because each track item panel is empty.
Click the *Open* button next to the *Default List Item Panel* in *Panel List Field* section to edit it.
14. Right-click and add a Dynamic Bitmap Field for the cover art.
 - a. Call it "Cover Art".
 - b. Make it 40 x 40 pixels and at the left side of the item, inside the gray of the background bitmap.
 - c. Set its Dynamic Data to be retrieved from the list. To do this, when selecting which Dynamic Data ID to select, expand "TrackList" and select "CoverArt" from the list of sub data.
 - d. Leave *List Index Scope* to "SelfIndex", since this means to retrieve data from the item itself.
 - e. Set *Use Default Data* for *Rotation Angle* field. Select value type *Integer* and fill in the value of 0.
 - f. Go to the Bitmaps tab and create a bitmap for the file "NoCoverArtListIcon.png". Use this as the *Invalid Bitmap*.
15. Next, add a *DynamicTextField* for the track title.
 - a. Call it "Title".
 - b. Make it cover basically the rest of the item next to the cover art field.
 - c. Set its *Default Text* to "Dynamic String" and *Dynamic Data* similarly to the cover art, but this time select the "Name" list sub data.



16. Go back to the page easily by pressing the button.
17. At this point, the list looks much better and presents all the data entered in the DDH Runtime file.

18. Move the “CD Track List” panel to back (right-click and then *Move to Back*), the “CD Title” panel after and “CD Icon” on top.
19. Make sure the “CD Icon” is in the Top Layer by right-clicking on “CD Title” and select both *End Middle Layer* and *Start Middle Layer*.
20. Make sure it becomes focused by adding both the “CD Track List” panel to traversal order, and the “Track List” field.
21. There is no way to reach this page at the moment, so we need to add that button in the CD Player that takes us to this screen.

Go back to the “CD Player” page.

22. Create a new panel called “Track List Button”. Set its area to cover the upper right corner next to the CD PLAYER title.
23. In this panel, create a new *ActiveControlField*, which will act as the button. Name the field “Track List”.
 - a. Set *Outline Bitmap* to “button-normal” (same button bitmap we use for the other buttons).
 - b. Put the button itself in the upper right corner, and make it the same size as the other buttons.
 - c. Go to the *Bitmaps* tab, and add the bitmap file “TrackListIcon.png”. Use this bitmap as the field bitmap on the button.
 - d. Set *Select Action* to issue an Internal Action Set Focus to the “CD Track List” page. Remember to check the *Save State in History* option, so that *Back* works.
24. Make this panel appear at the bottom of the page, by right-clicking and then *Move to Back*.
25. Also, make it possible to navigate to this button using the faceplate by modifying the control map for this page.
 - a. Add the “Track List” button field to traversal order within the “Track List Button” panel, but do *not* add the panel to the page traversal order.
 - b. Instead, select the page and edit its control map (*Edit Control Maps in Details*).
 - c. First time you click *Add* a new control map will be created for the page. Call it “CD Player”.
 - d. Click *Add* to add a mapping for *Pressed* event on “Up” key to do an Internal Action *Set Focus* to the “Track List Button” panel.
 - e. Once again click *Add* to add another mapping for *Pressed* event on “Down” key to do an Internal Action *Set Focus* to the “Buttons” panel. Do not fill in a specific field.

26. *Reserialize and Run*.

27. When trying this out, it is obvious that by clicking a track in the list, we should be switching to that track and take us back to the CD Player view. Return to the “CD Track List” page.
 - a. Select “Track List” field.

- b. Find the *Select Action* under *Select/Focus Action* in the *Base List Field* section of the list *Details* view.
 - c. Set *Select Action* to issue an External Action “SelectTrack” in DemoCDFU. This is a value action, which means it also passes a value to the FU. Set *Dynamic Action Value* to be retrieved from the “Index” sub-data in “Track List” in DemoCDFU.
 - d. Also add an Internal Action *Back (PopStack)* to the *Select Action*, so that after selecting a track, the HMI jumps back to the playing screen.
28. Furthermore, this page should behave the same as the “Settings Menu”. Go ahead and assign it the same Control Map, by selecting the page and set *Control Map* (in the *Details* view) to “Settings”.
 - a. This is now a misleading name for that Control Map. Rename it in the Control Maps tab to be called “Up Down Navigation”.
29. The next thing these pages need is a transition effect when going between them. Set up *Specific Page Transitions* in the *Visual Effects* tab.

An idea is to slide in the list while sliding out the CD Player, and vice versa when returning. By using *Specific Page Transitions*, it is possible to tailor exactly the way a transition between two pages shall look.
30. To add a scrollbar, we simply add the existing “Scrollbar” panel to this page.
31. Right-click and *Move to Back* to make it part of the same layer as the list in the transition effect.
32. You may need to adjust the size and position of the list to match the scrollbar.
 - a. If the item panel is wider than the width of the list, it will be possible to scroll the list horizontally as well. Since this is not desired for our track list, make sure the width of the item panel is less or equal to the list’s width. This is easily fixed by modifying/checking their *Areas* in the *Details* view.
33. To make the scrollbar work with this list, we need to set up its *External Scrollbar Data*. This has to be set up the same way as we did it for the “Settings Menu”, since that is the way the “Scrollbar” panel’s *Graphical Scale Field* has been set up.
 - a. Click *SET* on *External Scrollbar Data*.
 - b. Set up the *Vertical Data* in the *Scrollbar Pixel Data* group.
 - c. Set *Total Size Data Id* to “Scrollbar Max Position” in internal FU (255).
 - d. Set *Start Pixel* to “Scrollbar Start Position”.
 - e. Set *End Pixel* to “Scrollbar End Position”.

To make sure the scrollbar works correctly, check if the following details are set:

- For “Track List” (a panel list field): in *Base List Field* section of the details, set *External Scrollbar Data* in *List Properties*, as described above.
- For “CD Track List” (a panel): in *Panel* section of the details, set *External Scrollbar Data*. In *Row Layout Configuration* section of the details select *Enabled*, then select *Vertical* for *Row Type* and *Center* for *Alignment*.

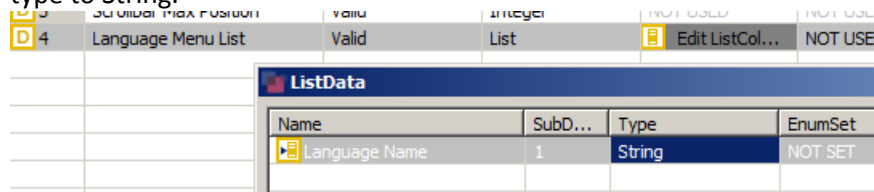
-
- Remember that we reuse the scrollbar, so we also have to use the identical dynamic data of the Internal FU.

34. *Target > Serialize and Run.*

10.6 EXERCISE C10.6 – ADD THE LANGUAGE MENU

From the items available in the Settings Menu, we will implement the possibility to change languages. We will use a Panel List Field to do this implementation, but instead of using Dynamic Data from an FU the Static Data Model will be used.

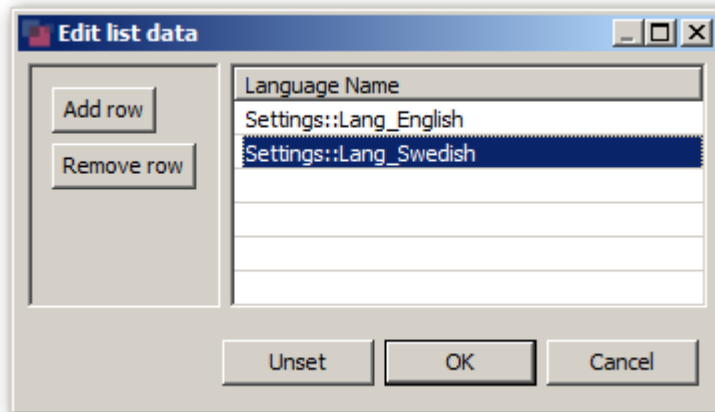
1. The submenu for “Set Language” is going to be a page of its own. Create a new “Settings Language” page.
 - a. Navigate to the *Pages* view, in Visual Model.
 - b. Right-click and select *New Page*.
 - c. Call the new Page “Settings Language”.
 - d. Click *EDIT* to edit it.
 - e. Set its background to the background bitmap we have used so far for all pages (“bg”).
 - f. Right-click and click *Add Existing Panel*, and add the Settings title panel. This is now the exact same panel as in the “Settings Main” page (that is, no copy).
 - g. Make “Settings” the new Page’s parent Profile.
2. For the appearance of the Menu Items, we want to create a new bitmap based on the one we created in exercise C10.2 (“Menu Item”). Go to the *Bitmaps* tab and make a copy of that bitmap. Check that bitmap file “panelmenuitem_focused.png” is selected for the Focused state bitmap. Call the new bitmap “Panel Menu Item”.
3. We need to set the static data that will be the base of the menu; this is done by creating a static List, where each list item will correspond to one menu item.
 - a. Open the FU class for the Internal FU in `\FU\Internal.fuclass`
 - b. Go to the *DynamicData* tab and add a new Dynamic Data called “Language Menu List” and set the type *List*.
 - c. Open the list sub data dialog by double-clicking in the *List Data* column.
 - d. Add a sub data called “Language Name” by clicking the *Add ListData* button and set type to *String*.



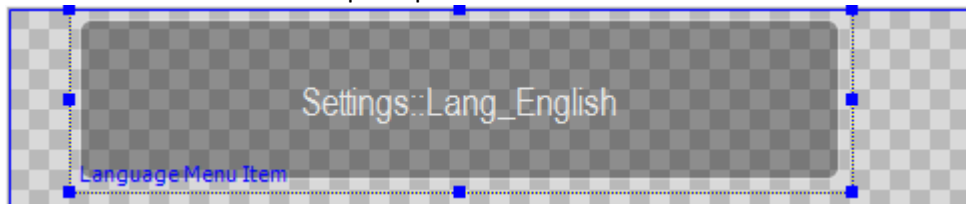
Name	SubD...	Type	EnumSet
Language Name	1	String	NOT SET

- e. Save and close the FU class.
4. We now need to specify which data we want to have in our Language Menu, this is done in the *Static Data Model*.
 - a. Go to the *Static Data* tab of the DDH file and select the Internal FU.
 - b. Double-click in the *Data* column for “Language Menu List” and add two rows to the list by pressing the *Add row* button.

- c. Set the data for each row to *Lang_English* and *Lang_Swedish*, then click *OK*.



5. Create a new panel for the language selection menu. It is going to contain a *Panel List Field* for the enumeration values available in the “Settings” FU.
- Right-click on the Page and click *Create New Panel*. Call it “Language Menu”.
 - Adjust the size of the panel so that it’s about the same as the “Settings Menu” panel in exercise C10.2.
 - Add a *PanelListField* to the panel called “Language Menu”.
 - This field shows up all red since we have not yet set any panel definitions yet. To do this we need to create a template panel that will be used for each menu item.



- Press the *Up* button until the top level of the visual model is shown and press the *Panels* button.
 - Right-click and select *New Panel...* Name the panel *Language Menu Item* and press *EDIT*.
 - Resize the panel to be the same width as the Language Menu and height of about 90 pixels.
 - Set the Background Bitmap to *Panel Menu Item*.
 - Add a Dynamic Text Field to the panel and call it “Language Name”.
 - For this new “Language Name” field set Dynamic Data to “*Language Menu List -> Language Name*”.
 - Set format string to CDPlayer, “Dynamic String”.
 - Set *Horizontal Alignment* to “Center”.
- e. Navigate back to the “Language Menu” by using the *Back* button and set the panel we just created as *Default List Item Panel* for the “Languages” field (in the section *Panel List Field* on the details pane).
- f. Set *List Data* to *Internal, Language Menu List*.

- g. Set *Row Type* to *Vertical*.
- h. Each item in the list should send a message make something happened in the system; this is done using *Item Specific Select Actions*.
- i. Add one action for each item and set *Actions* to trigger an *External Action* to set language with the item's corresponding index in the Settings FU. Choose the "SetDisplayLanguage" action and provide it the *Static Action Value* corresponding to the language. Also, set it to trigger an Internal Action *Back (PopStack)* so that there is a means of going back to the "Settings" menu with touch.

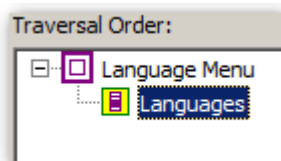
Item Specific Select Actions	
All items have Specific Select Action.	
Item Index	Actions
0	[Back(PopStack)] + [1 - Settings, 2 - SetDisplayLanguage: ValueAct
1	[Back(PopStack)] + [1 - Settings, 2 - SetDisplayLanguage: ValueAct

- j. Add the "CD Icon" panel to the page.

After this, your menu should look something like this:



- 6. Add this panel to Panel Traversal Order of the page and add the panel list field to Field Traversal Order.



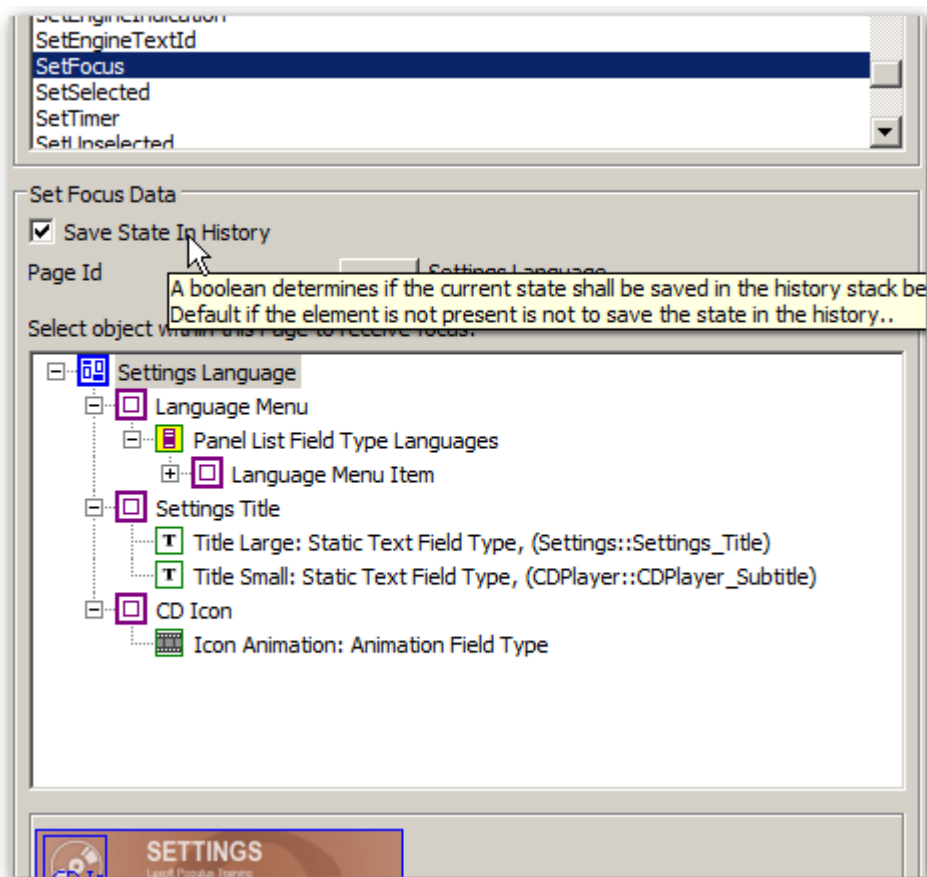
- 7. Move this panel backward in Z-order so that it is behind "Settings Title" (to match our layers in the transition effect).

8. Also, make sure “Settings Title” is starting the middle layer by right-clicking it and selecting “Start Middle Layer”.
9. Next, we want the Exit faceplate button to perform *Back (PopStack)* instead of releasing the whole profile. This should actually be the case throughout all pages in the HMI except for “Settings Main”. So setting up this control on the system-wide default control map makes sense. (The “Settings Main” page will override this behavior specifically for itself)
 - a. Select the page (“Settings Language”)
 - b. In the *Details* view, click on *Edit Control Maps* to bring up the control maps dialog.
 - c. In the top drop down of that window, select the Default control map (top item). With the default control map selected, click *Add* to add an entry to it.

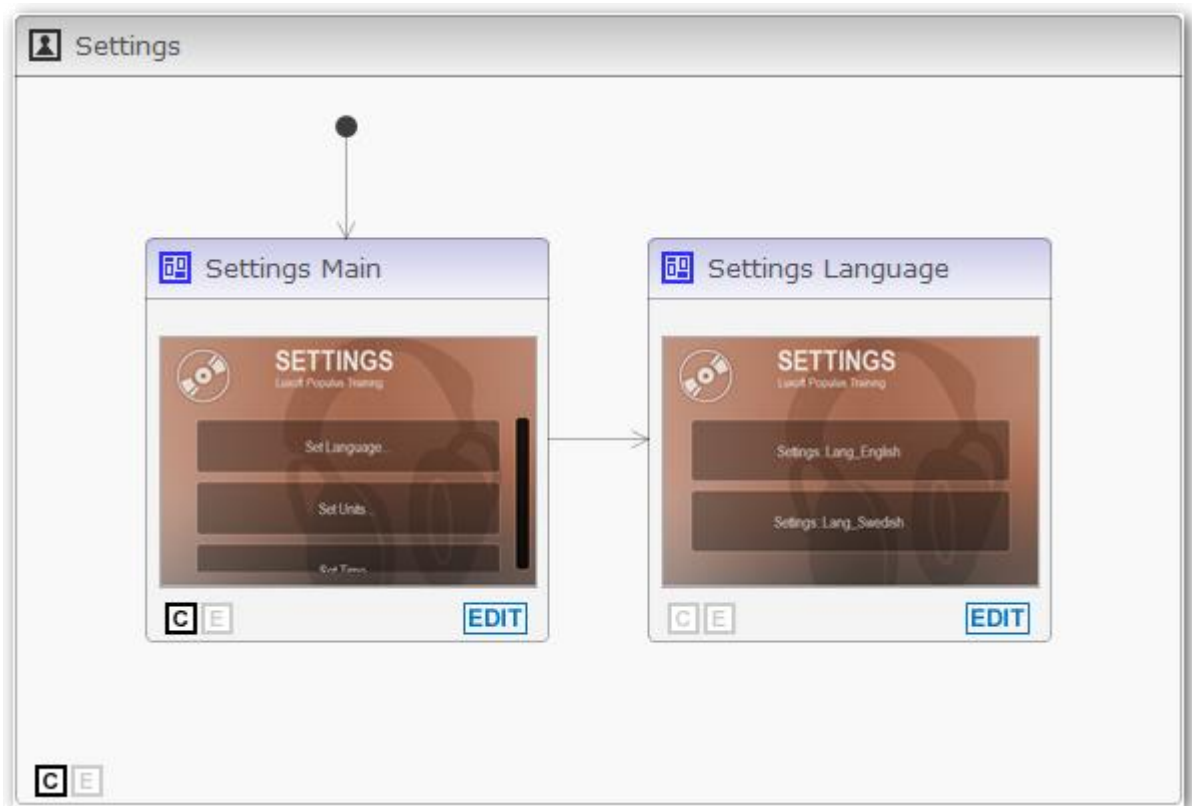
Default	5 - OK	NOT SET	NO	Pressed	NOT SET	[SetSelected] + []	NO
Default	5 - OK	NOT SET	NO	Released	NOT SET	[SetUnselected] + []	NO
Default	6 - Exit	NOT SET	NO	Pressed	NOT SET	[Back(PopStack)] + []	NO

10. Now it's time we could reach our new page with the language sub menu.

Go back to “Settings Main” page and the SetLanguage field. Set the *Select Action* to perform an Internal Action *Set Focus* to “Settings Language”. Don't select panel or field, but make sure *Save state in history* is checked (so that we can use *Back*).



11. The logic for our sub menu should now be finished, and will support both touch and physical keys. Have a look at the “Settings” profile:

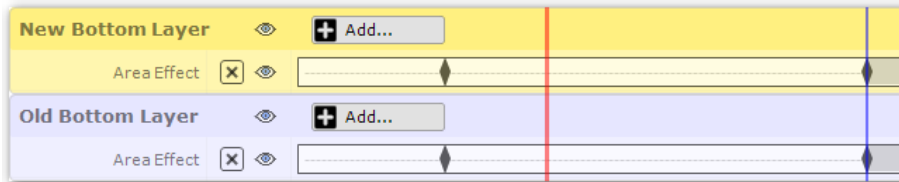


12. Serialize and run this HMI to make sure it works as intended, so far (*Target > Serialize and Run*).
13. Make the HMI more visually appealing by adding transition effects to going back and forth between the main menu and sub menu.

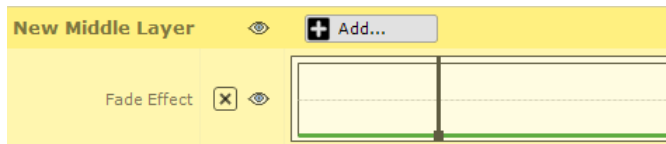
We are making two kinds of page changes when we go back and forth: *Set Focus* and *Back*. This means we can set up more generic schemes for transition effects in the “Settings” profile, which would be reused if we choose to add more sub menu pages.

- Go to the *Visual Effects* tab and *Change Page Transitions* sub tab.
- Add a new transition to be in effect inside the Settings profile only.
- Create the *Change Page* transition effect to slide out the old bottom layer and slide in the new bottom layer.

Set up preview pages:



The middle layer containing the “Settings Title” panel would be drawn twice (once for each page in the transition) if we don’t make one of them invisible. The same thing goes for the top layer, which contains the “CD Icon” panel:



Note: The old page will unsubscribe from any data it uses during the transition, which in our case means that the CD will not be running its animation for the old page. This means we want to put the fade effect on the old top layer.

- d. Create a Back transition effect to do the opposite with the bottom layers. Slide in from the other direction.

14. *Target > Serialize and Run.*

Try to change language and navigate back to the CD Player page either via touch or by pressing the face plate buttons.

10.7 EXERCISE C10.7 – PRIORITY STACK

This is a theoretical exercise, so just think this through.

Simulate the priority stack by “processing” the requests given to you. From the beginning the stack is empty. There is a default profile being granted the display, Profile 1.

Track the state of the stack after each priority request or release:

1. Request Profile 2: Priority Level 3, Stackable;
2. Request Profile 5: Priority Level 5, Non-Stackable;
3. Request Profile 6: Priority Level 4, Non-Stackable;
4. Request Profile 3: Priority Level 4, Stackable;
5. Request Profile 2: Priority Level 5, Stackable;
6. Request Profile 4: Priority Level 5, Non-Stackable, Clear Non Stackable;
7. Release Profile 3;
8. Request Profile 6: Priority Level 5, Stackable, Clear Non Stackable;
9. Release Profile 6.

What is displayed now?

You should have a screen where Profile 2 displayed.

11 FU IMPLEMENTATION

In this chapter we are going to learn how to implement an FU application. The application will communicate with the Engine via TCP/IP by using the supplied communication classes included in the FU SDK.

Prerequisites

CMake 2.8.x (<http://www.cmake.org/>)

Visual Studio 2012

The material needed and the finished results after having followed these exercises are bundled with the installation of Luxoft Populus. Please navigate to your installation folder,

```
C:\Program Files\Luxoft Populus 4.xx\Exercises\  
HMI\Education Exercises\CD Player C10.6
```

Note: beginning with release 4.46.0 examples and exercises can be installed to a user-defined directory. The default location is `c:\Users\<user name>\Luxoft Populus xx`.

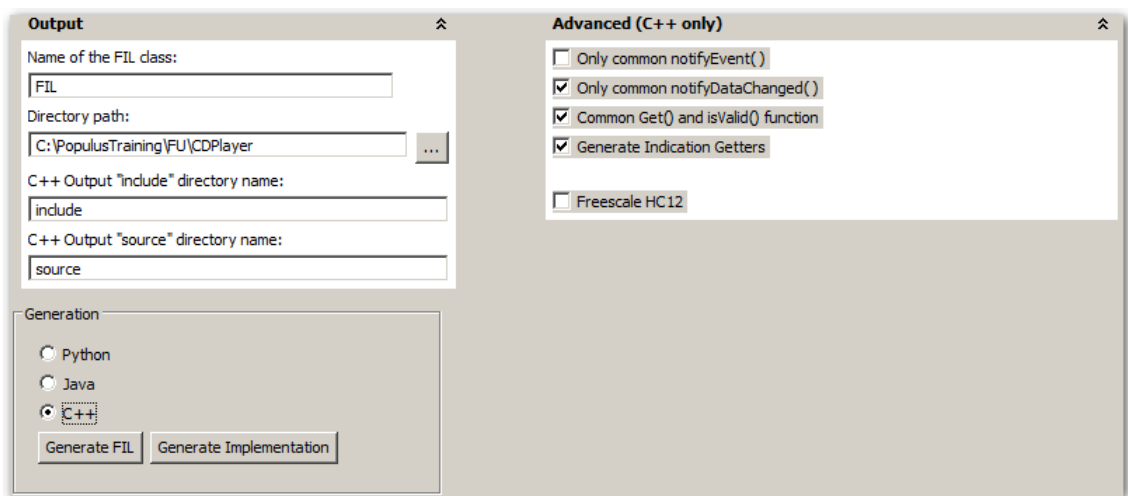
Within this folder there are sub folders with the end result after having finished each of the exercises in this chapter. From here, you can jump to any of the exercises and start off at that point by opening the project that corresponds to the previous exercise.

11.1 EXERCISE B11.1 – SIMPLE CD PLAYER STUB

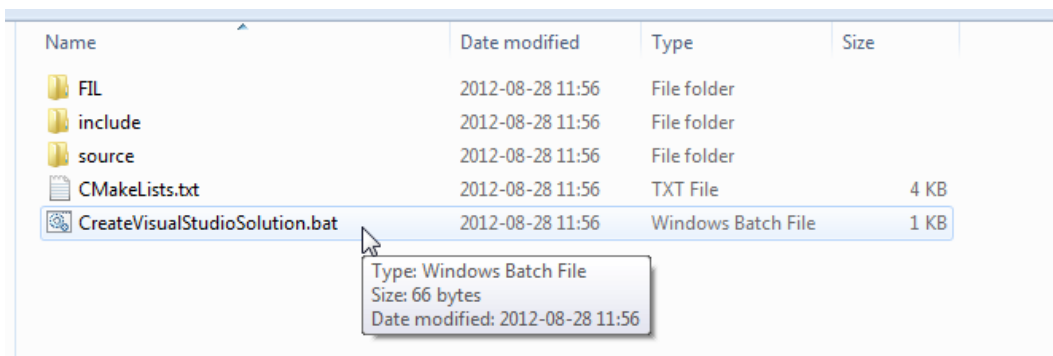
To get acquainted with the creation of Functional Units, we will create a simple implementation of the DemoCDFU interface we have been working with in the previous exercises. First start the Luxoft Populus Editor and open the finished result after exercise C10.6. You can use the prepared end result in the project “CD Player C10.6”. If you haven’t already, follow the instructions for how to import all the bundled HMI content in Chapter 4 - HMI CREATION BASICS.

1. Set the **CDPlayer.deploy** deployment as the current by right-clicking on it and selecting *Set as Active Deployment*. If you use a specific default.cfg, remember to change that also under *Target > Run Preferences*.
2. Create a new FIL Manifest for the CD Player implementation:
 - a. In the “FU” folder, right-click and select *New > Other...*
 - b. Select *FIL Manifest*.
 - c. Call it “CDPlayerFU.fil”. Make sure there are no spaces in the file name.
3. In the FIL Manifest editor, move over the DemoCDFU to the *Selected FUs* list to make this FIL include it.
4. Set the output directory in *Directory path* to a suitable working directory, e.g. “C:\PopulusTraining\FU\CDPlayer”. Remember to create it first in your file system.

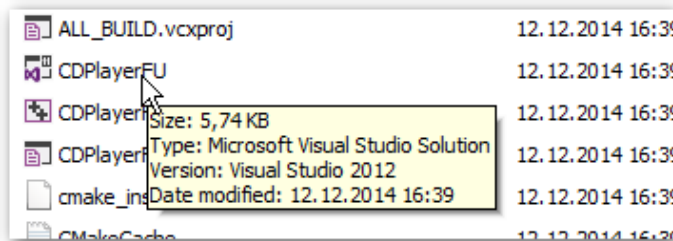
5. Leave the other settings as set by default:



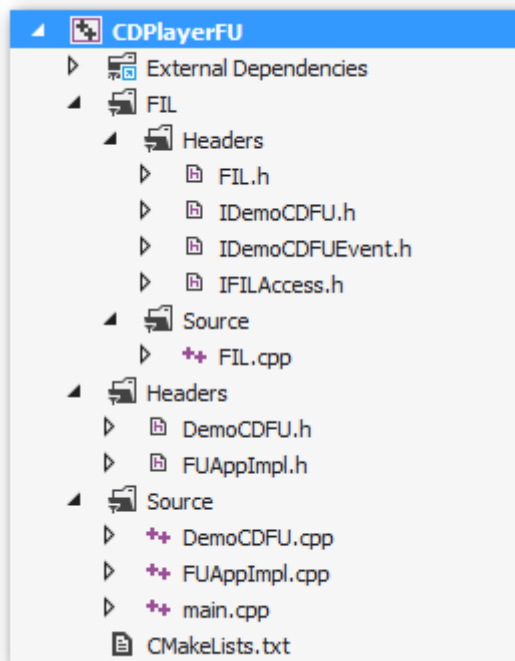
6. The implementation generation will create stub functions that return the static data set in the current DDH Runtime file at the time of generation. To utilize this feature, set the "SimulatedData.ddhruntime" file (or any other file you may have created) as the current runtime file.
7. Next, generate FIL and a stub implementation for it:
 - a. Click on *Generate FIL*;
 - b. Click on *Generate Implementation*.
8. In Windows Explorer, navigate to the output directory and run the *CreateVisualStudioSolution.bat* file. This will set up a Microsoft Visual Studio solution via CMake.



9. The output of the CMake generation is found in a subfolder called “build”. Open the Visual Studio solution file from there.



10. In Visual Studio, make sure the “CDPlayerFU” project is set as *Startup Project*.



These files are as follows:

Note: Never modify any of the generated files.

- a. *FIL.h* – Generated; Header for the FIL implementation;
- b. *IDemoCDFU.h* – Generated; Your FU Interface that needs to be subclassed;
- c. *IDemoCDFUEvent.h* – Generated; Your interface towards FIL for notifications;
- d. *IFILAccess.h* – Generated; Internal interface in FIL;
- e. *FIL.cpp* – Generated; Implementation of FIL for this node;
- f. *DemoCDFU.h* – Subclass implementation of IDemoCDFU.h;
- g. *FUAppImpl.h* – Subclass to the FUApp application helper in the FU SDK;
- h. *DemoCDFU.cpp* – FU implementation (stub implementation for now);
- i. *FUAppImpl.cpp* – FU application;
- j. *Main.cpp* – Executable entry point;
- k. *CMakeLists.txt* – Cmake configuration; use this to add/remove source files etc.

11. When running the FU for the first time, it will report that there are configuration entries missing. This is fine and the defaults (localhost etc.) will serve our purpose for now.

```
display_host_name_3 =  
ReadConfigFile, File [default.cfg] created/appended. Please edit and run again.  
display_port_nr_3 = default  
ReadConfigFile, File [default.cfg] created/appended. Please edit and run again.  
display_host_name_4 =  
ReadConfigFile, File [default.cfg] created/appended. Please edit and run again.  
display_port_nr_4 = default  
ReadConfigFile, File [default.cfg] created/appended. Please edit and run again.  
display_host_name_5 =  
ReadConfigFile, File [default.cfg] created/appended. Please edit and run again.  
display_port_nr_5 = default  
ReadConfigFile, File [default.cfg] created/appended. Please edit and run again.  
display_host_name_6 =  
ReadConfigFile, File [default.cfg] created/appended. Please edit and run again.  
display_port_nr_6 = default  
ReadConfigFile, File [default.cfg] created/appended. Please edit and run again.  
display_host_name_7 =  
ReadConfigFile, File [default.cfg] created/appended. Please edit and run again.  
display_port_nr_7 = default  
ReadConfigFile, File [default.cfg] created/appended. Please edit and run again.  
display_host_name_8 =  
ReadConfigFile, File [default.cfg] created/appended. Please edit and run again.  
display_port_nr_8 = default
```

12. With the FU up and running, let's start up the HMI and see that the FU works.
- Go back to the Luxoft Populus Editor.
 - Make sure that the FU Simulator is not trying to connect to the Engine, at least not with the DemoCDFU we have running.
 - Target > Run* (or *Target > Serialize and Run* if necessary).
13. The HMI will contain the data set in the DDH Runtime file at the time of FU Implementation generation:



14. Let's have a look at the inner workings of the FU code:

Modify the FU so that it is playing one of your favorite tracks by modifying the *Artist* and *Title* strings.

- a. Find the getter for the *Title*.

```
const std::wstring& DemoCDFU::getTrackTitle(bool& valid)
{
    valid = true;
    static std::wstring retval=L"Dancing Queen";
    return retval;
}
```

- b. Next, do the same for the *Artist*.

15. Add code to make the FU behave more like a real FU. By sending the *Play* action, the *Playing* indication should switch to *true*. By sending the *Pause* action, the *Playing* indication should switch to *false*.

- a. Add a private boolean member "*m_isPlaying*" to the DemoCDFU.h file:

```
18: private:
19:     IDemoCDFUEvent *m_pDemoCDFUEvent;
20:     bool m_isPlaying;
21:
```

- b. Set it to *false* in the constructor:

```
19: DemoCDFU::DemoCDFU(): m_isPlaying(false)
20: {
21: }
```

- c. In the *doPlayAction()*, set it to *true*.
d. In the *doPauseAction()*, set it to *false*.
e. In the indication getter, return *m_isPlaying* to the HMI.
f. Recompile and run the FU.

16. You should now see your chosen Artist and Title. But the play/pause functionality does not seem to work?

17. We forgot something important. We did not notify FIL about the change.

- a. Go back to the *doPlayAction()* and *doPauseAction()* and add a call to *notifyIndicationsChanged()*. Without this, FIL will not know it is time to call the getter for the indication and send new indications to the Engine.

```
m_isPlaying = false;
m_pDemoCDFUEvent->notifyIndicationsChanged();
```




By pressing “Play” at this stage, the CD animation starts spinning and the track info shows up:



11.2 EXERCISE B11.2 – IMPLEMENT SUPPORT FOR COVER ART

Sending bitmaps to the Engine differs a bit from sending numeric data and strings. There are two ways to send bitmaps to the Engine, of which we will try both; Binary and Path. Let's set it up with Path first, which is the simplest to get started with but requires the FU to access the same file system as the Engine (which we do on the PC).

1. We need to have a bitmap file to transfer first. There are suitable example bitmaps in the Exercise folder under "Materials" (\Exercises\FU\Education Exercises\Exercise B11\Material). Find all the "coverart*.jpg" files and copy them to our C:\PopulusTraining folder.
2. Add the path to "coverart0.jpg" as a static string in the class:
 - a. Go to the header file and declare a private `std::string` called `s_path`.

```
22
23     static std::string s_path; // Used for the static coverart path
24
25     ...
```

- b. Initialize it to "C:\\PopulusTraining\\coverart0.jpg" in the cpp file. We can hard code and send this string as-is, since its UTF-8 representation is the same.

```
23
24     std::string DemoCDFU::s_path = "c:\\PopulusTraining\\coverart0.jpg";
25
```

- c. Add the necessary information about the bitmap to the info getter (`getCurrentCoverArtInfo`). For a static bitmap like this one there is no need to use checksums, since the bitmap will never change. Just set this to zero. Make sure `dataType` is `BITMAP_PATH` and set the `totalSize` to the length of the path (the payload we are going to transfer over ODI). Set validity to true.

```
void DemoCDFU::getCurrentCoverArtInfo(U32& totalSize, U32& crc32, U8& dataType, bool& valid)
{
    valid = true;
    dataType = BITMAP_PATH;
    crc32 = 0;
    totalSize = s_path.length();
}
```

- d. The Engine is going to request the path chunk by chunk with the data getter (`getCurrentCoverArt`). Implement with `memcpy` and use the `c_str()` member function of `std::string` to access the bytes.

```
165 U32 DemoCDFU::getCurrentCoverArt(U32 offset, U8* buffer, U32 bufferSize)
166 {
167     U32 chunkSize = std::min(s_path.length() - offset, bufferSize);
168     memcpy(buffer, &s_path.c_str()[offset], chunkSize);
169     return chunkSize;
170 }
```

3. Recompile and run the FU. You should now see your bitmap appear as cover art when the CD Player HMI is playing.



4. Let's modify our code again, and change it to send the bitmap data itself, and not the path. This is the only option available if the Engine and the FU do not share file systems. However, it requires more bandwidth since the file size is considerably larger than its path.
 - a. First, we will read the bitmap file from disk and store it in a buffer in the constructor. This is just to simplify things and have the bitmap ready in memory. In a real FU, you would probably choose a smarter way to handle your bitmaps.

Add a buffer for the bitmap, "m_buffer", to the header file as a `std::vector<U8>`.

```
26 |         std::vector<U8> m_buffer; // Coverart buffer
27 |
```

Do not forget to add statement in DemoCDFU.h: `#include <vector>`.

- b. In the constructor (to make things easy), read the bitmap file and store its contents in the buffer.

```
21 | DemoCDFU::DemoCDFU(): m_isPlaying(false)
22 | {
23 |     std::ifstream bitmapFile(s_path.c_str(), std::ios::binary);
24 |     m_buffer.assign((std::istreambuf_iterator<char> (bitmapFile)),
25 |                    (std::istreambuf_iterator<char>()));
26 | }
27 |
```

Do not forget to add statements in DemoCDFU.cpp: `#include <fstream>` and `#include <iterator>`.

- c. Now the getters need to be modified. First, we need to change the `getCurrentCoverArtInfo()` to return the size of the bitmap data and not the path.

```
void DemoCDFU::getCurrentCoverArtInfo(U32& totalSize, U32& crc32, U8& dataType, bool& valid)
{
    valid = true;
    dataType = BITMAP_BIN;
    crc32 = 0;
    totalSize = m_buffer.size();
}
```

- d. Next, the retrieval of the bitmap data instead of the path needs to be implemented in `getCurrentCoverArt()`. (Basically just change `s_path.length()` to `m_buffer.size()`).

```
171 U32 DemoCDFU::getCurrentCoverArt(U32 offset, U8* buffer, U32 bufferSize)
172 {
173     if (offset < m_buffer.size())
174     {
175         U32 chunkSize = std::min(m_buffer.size() - offset, bufferSize);
176         memcpy(buffer, &m_buffer[offset], chunkSize);
177         return chunkSize;
178     }
179     else
180     {
181         return 0;
182     }
183 }
```

- e. We could change the validity info as well, to only send a bitmap if we loaded it successfully. Modify `getCurrentCoverArtInfo()`.

```
void DemoCDFU::getCurrentCoverArtInfo(U32& totalSize, U32& crc32, U8& dataType, bool& valid)
{
    valid = !m_buffer.empty();
    dataType = BITMAP_BIN;
    crc32 = 0;
    totalSize = m_buffer.size();
}
```

5. Recompile and run. The same bitmap will now show up in the HMI when playing.

11.3 EXERCISE B11.3 – IMPLEMENT TRACK LIST

In the HMI, there is the possibility to view all available tracks on the CD by clicking on the button at the top right in the display. This brings up a panel in the HMI which contains a `PanelListField`, subscribing to the data `TrackList` in our FU.



The `PanelListField` has been set up to display the sub data for track title (*Name*) and the sub data for cover art (*CoverArt*), for each item in the list.

The Implementation generation in the Editor has given us a static list to work with, containing all the information as static data. To learn how to work with lists, we are going to implement our FU so that it is possible to play each of the tracks in the list.

If your generation did not include a static list (your DDH Runtime file did not specify any) then go back and regenerate an implementation after having added about five items to the list.

Warning! When you regenerate, be sure to specify a temporary output folder, otherwise the Editor will overwrite what you have done so far.

We are going to make it so that the currently playing track is displayed in square brackets [...] and enable the “Previous” and “Next” buttons to traverse within the tracks in the list (also updating the information about the currently playing track).

1. Add a variable of the type *int* to track the currently playing track index. Call it “*m_trackNumber*”. Initialize it to zero in the constructor.
2. Make it so that the *doPreviousAction()* and *doNextAction()* action handlers decrement and increment this variable. Now, when we change tracks we will also change the current track title, artist, track number and cover art data values, so FIL needs to be notified. Make a helper function (private) that updates track number and notifies FIL about all the data that has changed, that we can call from multiple places.
3. Update the getters for the title and track number. Modify the title getter so that it retrieves this information from the track list. (You may put track names that make sense into the static list if you wish.)
4. Update the “next/previous track available” indications to be enabled at all times or optionally add proper conditions for enabling and disabling them based on navigation

availability in the list. The indication changes are signaled with the *notifyDataChanged* method that has additional parameter specifying which data has changed.

5. Recompile and run to verify the behavior.
6. In the list, the currently playing track should be displayed using square brackets. For the sub data *Name* getter, include these brackets in the text returned if this is the currently playing track. This is implemented in *getTrackList_NameAt*.
7. Whenever the currently playing track is changed, we also need to update the list. We could of course notify FIL that the entire list has changed, but this is unnecessary since we only changed two items; the previously playing track and the currently playing track. Update the helper function we created for this purpose.
8. Also, make the *doSelectTrackAction()* so that it changes the track to the supplied number. This is called when the user selects a track in the list.
9. Recompile and run, to make sure this works.
10. Add support for showing cover art in the track list. There are multiple “coverart*.jpg” files in the “Material” folder that we copied to *C:\PopulusTraining* in a previous exercise. Modify the track list so that each track contains a buffer for the cover art, and read all cover art bitmap files into the corresponding buffer at startup (constructor).
 - a. Update the cover art getter in the list to provide the bitmap buffer for the corresponding track item.
 - b. Since we now have a bitmap corresponding to each track, we can remove the *m_buffer* for the current playing track and make the getter for the current cover art image return the one in the track list for the current track number. (A shortcut here is to just call the getter for the list with *m_trackNumber* as index parameter.)
 - c. Add *include <sstream>*.
 - d. Add *include "crc32static.h"* (from the fu-sdk).
 - e. Use the namespace *ODI_UTILS_NS*.
 - f. When retrieving the cover art info always set valid to true: *getCurrentCoverArtInfo*, *getTrackList_CoverArtInfoAt* and *getTrackList_NameAt*.
 - g. Do not forget to change the path to the “coverart*.jpg” files.



11. Recompile and run.