

# Floorplan Manager for ABAP - Developer's Guide



**Release 701**



## Copyright

© Copyright 2008 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group. Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.






Java is a registered trademark of Sun Microsystems, Inc

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, xApps, xApp, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

## Icons in Body Text

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help → General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

## Typographic Conventions

Type Style	Description
<i>Example text</i>	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options.  Cross-references to other documentation.
<b>Example text</b>	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
Example text	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
<b>Example text</b>	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
EXAMPLE TEXT	Keys on the keyboard, for example, F2 or ENTER.

Floorplan Manager.....	6
Getting Started.....	6
User Interface Building Blocks.....	7
IF_FPM_UI_BUILDING_BLOCK Interface.....	7
Creating a Simple FPM Application.....	9
Creating a Web Dynpro Component.....	10
Creating a Web Dynpro Application.....	11
Using Application Parameters.....	12
Creating an Application Configuration with the FPM Configuration Editor.....	13
Testing your FPM Application.....	16
Configuration Editor for Floorplan Manager.....	16
Form Editor for Floorplan Manager.....	19
List Editor for Floorplan Manager.....	21
Tabbed Component Editor for Floorplan Manager.....	22
Design Time with the FPM Configuration Editor.....	23
Floorplan Instances in the FPM Configuration Editor.....	23
Adding and Activating Sub-Steps.....	25
FPM Toolbar.....	26
Toolbar Buttons.....	27
IF_FPM_CNR_GAF Interface.....	31
IF_FPM_CNR_OIF Interface.....	34
FPM Identification Region (IDR).....	36
IF_FPM_IDR Interface.....	37
Providing a Link to the FPM Configuration Editor in the IDR.....	40
Quick Help.....	40
Create Quick Help.....	41
Variants.....	43
Initial Screen.....	44
Confirmation Screen.....	45
FPM Event Loop.....	46
Different Categories of Web Dynpro Interfaces.....	49
Generic User Interface Building Block (GUIBB).....	50
Feeder Classes.....	50
Form Component (GUIBB FORM).....	51
IF_FPM_GUIBB_FORM Interface.....	52
Form Editor for Floorplan Manager.....	56
Add Form.....	58
List Component (GUIBB LIST).....	60
IF_FPM_GUIBB_LIST Interface.....	61
List Editor for Floorplan Manager.....	65
Add List.....	66
Additional Information on the List Component.....	68
Tabbed Component (GUIBB TABBED COMPONENT).....	70
Tabbed Component Editor for Floorplan Manager.....	70
Add Tabbed Component.....	71
Changing the Tabbed Component Dynamically at Runtime.....	73
Navigation.....	74
Launchpad.....	75

Create a Launchpad with Applications.....	77
Include a Launchpad in the User Interface .....	77
Working in the Navigation Customizing .....	78
IF_FPM_NAVIGATION API (Runtime class CL_FPM_NAVIGATION) ...	81
Integration: Navigation in the Event Loop .....	86
IF_FPM_NAVIGATE_TO API .....	86
Suspend and Resume .....	88
Handling Dialog Boxes .....	90
Triggering a Data-Loss Dialog Box in the FPM Event Loop.....	90
Handling Application-Specific Dialog Boxes .....	91
IF_FPM_WORK_PROTECTION Interface.....	94
FPM Message Management .....	95
Using the FPM Message Manager .....	96
IF_FPM_MESSAGE_MANAGER Interface .....	97
Methods for Reporting Messages .....	97
Methods for Raising Exception Messages .....	103
Method for Clearing Messages .....	105
FPM Message Manager FAQ .....	106
Handling of Transactions.....	107
IF_FPM_TRANSACTION Interface .....	108
Resource Management .....	109
Setting the Transient Flag.....	113
Using IF_FPM_RESOURCE_MANAGER to Veto Release Decision ...	113
Using an FPM Application Controller.....	114
IF_FPM_APP_CONTROLLER Interface.....	115
Using an Application-Specific Configuration Controller .....	115
Sharing Data between UIBBs from different Components .....	119
Embedding and FPM Application .....	120



## Floorplan Manager

Floorplan Manager (FPM) is a Web Dynpro ABAP application that provides a framework for developing new Web Dynpro ABAP application interfaces consistent with SAP UI guidelines. FPM currently supports you in creating and configuring user interfaces with the following floorplans:

- Object Instance Floorplan (OIF)
- Guided Activity Floorplan (GAF)
- Quick Activity Floorplan (QAF)

The following floorplan areas can be configured using the FPM configuration editor:

- Identification Region (IDR)
- Message Region (MR)
- Context Navigation Region (CNR)
- Roadmap Element

Floorplan content areas must also be UI guideline compliant and FPM provides pre-defined UIBBs to support you in creating and configuring application-specific views (“freestyle areas”). The common UI patterns Form, List, and Tabbed Area can be configured using the FPM configuration editor.

FPM includes APIs for common functions such as navigation, data-loss handling, messaging, and personalization.

FPM allows for modification-free customer adaptations.

### System Requirements

This document outlines the features of Floorplan Manager as of release SAP NW 7.0 Enhancement Package 1 and SAP NW 7.1 Enhancement Package 1.



## Getting Started

This section provides you with an overview of an FPM application and the steps required by you to create a simple *Hello World* example application.

Once you have created your application, you are introduced to the FPM Configuration Editor, which allows you to edit your application and to configure it at design time.

The FPM event loop and its various activities are explained to you, and finally you are presented with time-saving design templates, allowing you to create guideline compliant user-interfaces.

## Assumptions

A knowledge of ABAP OO and Web Dynpro for ABAP is assumed.



## User Interface Building Blocks

From an FPM perspective, UIBBs are the interface views (Web Dynpro ABAP windows) that are provided by the external application and not by FPM itself.

In order that the FPM framework recognizes a UIBB, the Web Dynpro component that provides the UIBB must implement the `IF_FPM_UI_BUILDING_BLOCK` Web Dynpro interface. The `IF_FPM_UI_BUILDING_BLOCK` interface ensures that the Web Dynpro application can take part in the FPM event loop.

For more information, see [IF\\_FPM\\_BUILDING\\_BLOCK Interface](#).




## IF\_FPM\_UI\_BUILDING\_BLOCK Interface

This Web Dynpro interface ensures that a Web Dynpro application and its UIBBs can take part in the FPM Event Loop.

### Methods

The methods of this interface are described in the following table:

Method Name	Method Description
FLUSH	<p>This is the first method called after the FPM event loop has been started.</p> <p>In this method, the UIBB needs to transport all modified data from the views to other components the UIBB wants to communicate with later on.</p> <p>Normally this data transport is done automatically using Web Dynpro context mapping. Therefore, you will only need to do a specific implementation of this method if you are not using these automatic mechanisms.</p>
NEEDS_CONFIRMATION	<p>With this method, the UIBB requests that the subsequent event processing is stopped and asks the user for confirmation by way of a dialog box. Depending on the action the user takes in the dialog box, the event loop is continued or cancelled. For more details, refer to chapter Triggering a Data Loss Dialog Box.</p>
PROCESS_EVENT	<p>Within this method the UIBB completes the following tasks:</p>

Method Name	Method Description
	<ul style="list-style-type: none"> <li>• Checks for local consistency (validation, missing data, etc).</li> <li>• Perform the actual event processing.</li> </ul> <p>The local check is needed to inform the user of potential input errors as soon as possible. In accordance with UX guidelines, checks are to be performed continually (as long as they are not too performance-intensive). For example, when switching from one view to another view in an OIF application, the view (UIBB) which is moved away from must check for local consistency.</p> <p>However, this does not exempt the application from performing a complete check (including performance critical checks) before saving. This must be handled in the method <code>IF_FPM_TRANSACTION_CHECK_BEFORE_SAVE</code>.</p> <p>Besides the consistency check this method contains the actual processing of the event. For this, the current event can be identified through the attributes <code>MV_EVENT_ID</code> and <code>MO_EVENT_DATA</code> on the passed on event instance <code>io_event</code>. Depending on whether the event is processed successfully or not, the exporting parameter <code>EV_RETURN</code> must be filled with either <code>IF_FPM_CONSTANTS~GC_EVENT_RESULT-OK</code> or <code>IF_FPM_CONSTANTS~GC_EVENT_RESULT-FAILED</code>.</p> <p>A typical implementation of <code>PROCESS_EVENT</code> is shown below:</p> <p> <b>Syntax</b></p> <pre> 1. IF io_event-&gt;mv_event_is_validating =    abap_true. 2.   Do local checks and report messages    if needed 3.   ENDIF 4.   CASE io_event-&gt;mv_event_id. 5.     WHEN XYZ 6.       Handle event and fill EV_RETURN        accordingly with a value from        IF_FPM_CONSTANTS~GC_EVENT_RESULT 7.   ENDCASE.</pre> <p>If the event processing requires further user interaction (e.g. asking for further data in a dialog box), the event processing can be deferred by returning <code>EV_RETURN = IF_FPM_CONSTANTS~GC_EVENT_RESULT-DEFER</code>.</p>
AFTER_FAILED_EVENT	This method is called by the FPM if an event could not be processed successfully. In this case the UIBB needs to



Method Name	Method Description
	<p>ensure that its UI reverts to the state before the user interaction occurred.</p> <p>Example:</p> <p>Selecting an option in a 'Lead' field in a table triggers the display of the details of a new line in another UIBB. The event could fail if the UIBB for the details contains unsaved data for the previously selected table line. As the detail form still contains the details of the original table line (after the failed event), the Lead selection must be reverted to the original table line too.</p> <p>If the <code>PROCESS_EVENT</code> method of the current UIBB has been processed successfully, but the event processing failed due to a problem in another UIBB, the actual event processing needs to be reverted as well. The parameter <code>IV_REVERT</code> indicates this situation.</p>
<code>PROCESS_BEFORE_OUTPUT</code>	<p>The last method to be called on the UIBB is the <code>PROCESS_BEFORE_OUTPUT</code>. The data to be displayed is read from the model.</p>



## Creating a Simple FPM Application

Here you create a simple *Hello World* FPM application based on either the OIF or GAF. The OIF application will contain 2 tabs, each containing a single subview tab; the GAF application will contain 2 road steps.

This process is performed in the *Web Dynpro ABAP Workbench*.

### Process

You construct an FPM application by completing the following steps:

1. [Create a Web Dynpro Component](#) with the required UIBBs and implement the Web Dynpro interface `IF_FPM_UI_BUILDING_BLOCK`.
2. [Create a Web Dynpro Application](#) and specify parameters according to which floorplan instance you are using.
3. Using the *FPM Configuration Editor*, [create a configuration for the application](#).
4. [Test your application](#).

An FPM application is composed of a number of different Web Dynpro components (most of which are instantiated dynamically at runtime). However, the following two components are always present:



- a floorplan-specific component (FPM\_GAF\_COMPONENT or FPM\_OIF\_COMPONENT)
- a component for the Header Area (FPM\_IDR\_COMPONENT)

In simple terms, the configuration of an FPM application is the configuration of these two components.

## Creating a Web Dynpro Component

### Procedure

#### Creating the Web Dynpro Component

1. Open the *Web Dynpro ABAP Workbench*.
2. In the *Object Navigator*, right-click the Web Dynpro node and choose  *Create* → *Web Dynpro Component (Interface)* .
3. In the *Web Dynpro: Component/Create Interface* dialog box, enter a name, description and window name (the window name must be different from the View name).
4. Save your entry.
5. In the *Attributes* section view of the *Create Object Entry Directory* dialog box, enter the relevant Package.
6. Save your entry. The preview displays your new (inactive) Web Dynpro Component.
7. Choose the *Implemented Interfaces* tab.
8. In the first row of the Name column, enter the FPM interface IF\_FPM\_UI\_BUILDING\_BLOCK and save your entry.
9. In the *Action* column, choose *Reimplement*. The icon in the *Implementation State* column indicates that your component is completely implemented.
10. Choose *Activate*.
11. In the *Activation* dialog box, select all associated, inactive components and choose *OK*.

#### Adding Views to your Web Dynpro Component

When you create a component, Web Dynpro automatically creates and assigns a Window and a View to it. You may add further Windows and Views. It is recommended that you add only one View to one Window.

1. In the Object Navigator, find your new Web Dynpro component and expand its node.
  1. Expand the Views node and double-click the existing View. The View appears in the preview.
  2. In the *Layout* tab, click once on the Caption element. A blue square appears in the preview, ready to display your text.
  3. In the *Properties Section*, enter `He11o` in the *Text* property. Choose *Save* and your text appears in the preview.
2. Choose *Activate*.
  1. In the Activation dialog box, select all associated, inactive components and choose OK.
3. Add a second View:
  1. Right-click the *View* node and choose *Create*. Give your View a name and choose OK.
  2. Add a caption element and enter the text `Welcome to the world of FPM.`
4. Add this view to a new Window (which you create now):
  1. Right-click the *Windows* node and choose *Create*.
  2. In the *Web Dynpro: Create Window* dialog box, enter a Window name and choose *OK*.
  3. The preview automatically displays the *Window* tab. In the *Window Structur* column, there is a node with your new Window's name.
  4. Drag your new View from the *Object Navigator* onto this node so that it is included in the *Window* structure (expand the node to see the new listed below it).
  5. Save and activate your new Window.

## Result

You have now created a Web Dynpro Component, implemented the required `IF_FPM_UI_BUILDING_BLOCK` interface and configured two views (in two separate windows) for your component.



## Creating a Web Dynpro Application

### Prerequisites

You have already created a Web Dynpro component with two views.

### Procedure

1. In the *Object Navigator*, right-click your Web Dynpro Component and choose **► Create → Web Dynpro Application ◀**

In the *Create Web Dynpro Application* dialog box, enter a name for your application and choose *OK*. Your new Web Dynpro Application appears in the preview.

2. Enter the following information to create either an OIF or a GAF application:

- *Component:* FPM\_OIF\_COMPONENT / FPM\_GAF\_COMPONENT
- *Interface View:* FPM\_WINDOW
- *Plug Name:* Default

3. Save your entries.

In the *Create Object Directory Entry* dialog box, enter the relevant Package and choose *OK*.

## Result

You have created a Web Dynpro application based on an OIF or GAF floorplan instance.

If you want to add parameters to your application, [see Using Application Parameters.](#)



## Using Application Parameters

Application parameters are defined at Web Dynpro Application level.

To define your application parameters, proceed as follows:

1. In the *Web Dynpro Object Navigator*, double-click your Web Dynpro application.
2. Choose *Parameters*. You can add arbitrary parameters as application-specific attributes to your Web Dynpro application. During runtime, these parameters are exposed via `IF_FPM->MO_APP_PARAMETER`. `MO_APP_PARAMETER` stores an instance of `IF_FPM_PARAMETER`. With this interface you are able to retrieve the parameters.

Note that there is no concept of mandatory or optional parameters. For security reasons, you must never trust parameters passed by a different application. Always complete a proper validation before you use application parameters.

There are other FPM-specific parameters which you can add to your application. These are detailed in the table below.

Parameter	Parameter Description
FPM_SHOW_MESSAGE_LOG	You can turn on a log history of the messages for a

Parameter	Parameter Description
	particular application. When the message log is turned on, all the previously reported messages are displayed.
FPM_MAXIMUM_MESSAGE_SIZE	When a message is created in the application, the message area displays as many messages as possible. As soon as the visible number of messages in the message area exceeds the configured message size, a scroll bar will appear in the message area, allowing the user to read all messages. The maximum size of the message is set via configuration.
FPM_HIDE_CLOSE	With this parameter, you can hide the <i>Close</i> button on the FPM toolbar for your application.



## Creating an Application Configuration with the FPM Configuration Editor

### Prerequisites

You have already created a Web Dynpro component with two views and have created a Web Dynpro application implementing the `FPM_OIF_COMPONENT` or `FPM_GAF_COMPONENT` interface.

### Procedure

1. In the *Object Navigator*, right-click your new Web Dynpro Application and choose *Create/Change Configuration*. The [FPM Configuration Editor](#) (Editor for the Web Dynpro ABAP Application Configuration) opens in a browser window.
2. Enter a name for your application's configuration in the *Configuration ID* field. Note that configuration names are global; you may not use the same configuration name for different applications.
3. Choose *Create*. In the *Create Configuration* dialog box, enter the relevant *Package* and choose *OK*.
4. The application configuration window displays your new configuration. Within your configuration are the following two components:
  - `FPM_OIF_COMPONENT` (OR `FPM_GAF_COMPONENT`)
  - `FPM_IDR_COMPONENT`
5. You will create configurations for both of these components. In the configuration column, enter names for both components and choose *Save*. A message appears to inform you that the components are saved, but that the configurations do not actually exist. You will create a configuration for the OIF (or GAF) component now.

6. Select the row containing your `FPM_OIF_COMPONENT` (OR `FPM_GAF_COMPONENT`) and choose *Go To Component Configuration*.
  1. Choose *Create* to configure the component.
  2. In the *Create Configuration* dialog box, choose the relevant *Package* and choose *OK*.
7. The *FPM Configuration Editor* displays the *Component Configuration* window for your OIF (or GAF) component. The *FPM Component Configuration* window is divided into the following areas:
  - *Navigation hierarchy*: shows the screen elements in your application which you can configure
  - *Preview*: displays the element you have selected in the hierarchy and allows you to change the attributes of the element
  - *Action area*: allows you to add various elements to your individual screens (for example, toolbar buttons, main views or UIBBs)

For a simple application, you require only one variant, one main view and one subview. The *FPM Configuration Editor* automatically provides these entities (with default IDs and names).
8. Complete the configuration by performing the following steps below.

#### **Configuring the Component and IDR Configurations**

##### **Configuring the `FPM_OIF_COMPONENT`**

1. The preview of the *Component Configuration* window displays 1 main view containing 1 subview.
2. To add the second main view tab, choose *Add Main View* in the action area.
3. In the hierarchy, expand the two *Main View* nodes and the two *Subview* nodes. Note the two UIBB elements, one for each subview. Choose the UIBB element belonging to the first subview to display its attributes in the preview.
4. Set these attributes to your first window (with accompanying view) of your Web Dynpro component (containing the text 'Hello').
  1. Enter the *Component name* (use the input help and search function to find your component).
  2. Enter the *View* (once you have entered the component name, the *View* input help displays the list of views for that component).
5. In the hierarchy, choose the other UIBB element to display its attributes. Set these attributes to your second window (with accompanying view) of your Web Dynpro component (containing the text 'Welcome to the world of FPM').
6. Choose *Save*. A confirmation (or error) message appears near the top of the screen.

You have now added your component views to the application and are ready to configure the IDR Component of your application's configuration.

#### Configuring the FPM\_GAF\_COMPONENT

1. The preview of the *Component Configuration* window displays 1 main step containing 1 UIBB. There are also two buttons, *Previous* and *Next*, which the FPM automatically displays in the toolbar for you.
2. To add the second step, choose *Add Main Step* in the action area.
3. In the hierarchy, expand the two *Main Step* nodes. Note the two UIBBs, one for each step. Choose the first UIBB element to display its attributes in the preview.
4. Set these attributes to your first window (with accompanying view) of your Web Dynpro component (containing the text 'Hello').
  1. Enter the *Component name* (use the input help and search function to find your component).
  2. Enter the *View* (once you have entered the component name, the *View* input help displays the list of views for that component).
5. In the hierarchy, choose the other UIBB element to display its attributes. Set these attributes to your second window (with accompanying view) of your Web Dynpro component.
6. Choose *Save*. A confirmation (or error) message appears near the top of the screen.

You have now added your component views to the application. You are now ready to configure the IDR Component of your application's configuration.

#### Configuring the FPM\_IDR\_COMPONENT

Once you have created a configuration for your OIF (or GAF) component, you are then ready to create a configuration for the IDR component.

1. In the action region of the *Component Configuration* window, choose *Configure IDR*. The *Configuration ID* field displays the name you provided in the previous steps for your IDR component configuration.
2. Choose *Create*. In the *Create Configuration* dialog box, enter the relevant *Package* and choose *OK*. The *Component Configuration* window displays your IDR Configuration.
3. In the hierarchy, choose *IDR Basic*. The preview displays the attributes of the IDR Basic. Enter the following data:
  - *Application Title*
  - *Tooltip* (optional)

4. Choose *Save*. If there are error messages, they appear at the top of the window, underneath the window's title. Note that in an OIF application there is an extra button in the Action Pane, *Add IDR Extended*. This provides you with the optional *Extended Identification Region* and its attributes.

## Result

You have now created your first FPM application configuration. You can now test your FPM application.



## Testing your FPM Application

### Procedure

1. Open the *Web Dynpro ABAP Workbench*.
2. In the *Object Navigator*, locate your FPM application under the *Web Dynpro Applications* node.
3. Right-click your application and choose *Test*. Your application opens.

Note that your own component views (UIBBs) appear in the freestyle Content Area of the FPM application.



## Configuration Editor for Floorplan Manager

You use the Floorplan Manager configuration editor to enhance application user interfaces and fit them to your business needs.

### Features

The configuration editor consists of the following work areas:

- Navigation region

This region is divided into the following subregions:

- Control area

In this area, you select which screens you would like to configure for the selected Web Dynpro application. You can choose whether you want to see the preview of the initial screen, the main views of an application variant, or the confirmation screens of the selected application.

In this region, you can use the *Change* or *Display* buttons to display or configure the application's [global settings](#) and variant parameters.



Note



You can store multiple variants of a selected floorplan for one Floorplan Manager application. A variant gives you an additional level of differentiation within Floorplan Manager. For example, you can use variants to show multiple user roles in the same application at the same time. The individual variants are separated from one another in an initial screen.

- Hierarchy

This region gives you a hierarchical display of the elements you can configure. The elements you can configure depends on the current configuration of the application. The hierarchy shows elements on the screen that you have selected.

- Preview

The preview function shows you the user interface of the application. You can use the preview function to navigate within the user interface. However, not every element can be accessed. A selected element is highlighted in color in the hierarchy view and its attributes displayed in the attribute view.

- Action area

The action area contains links to all the actions you can execute for the selected application user interface. The actions that can be selected depend on the concrete configuration of the application. This means that selection of actions can differ within a configuration.

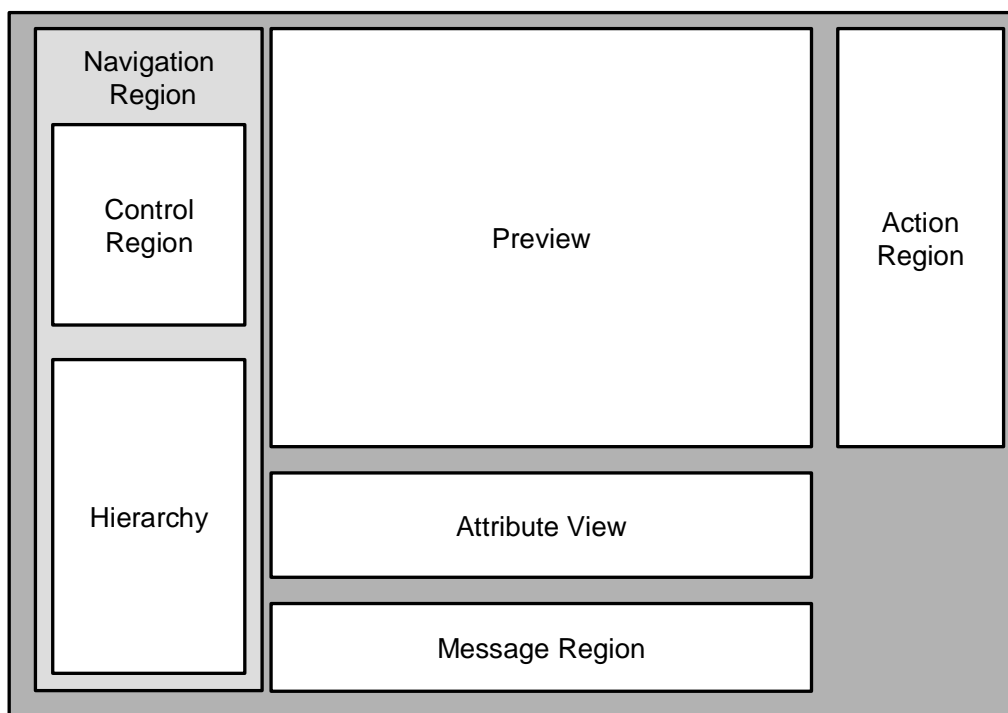
- Attribute view

When you select a configurable user interface element either in the preview or in the hierarchy, the attributes of these user interface elements are displayed in the attribute view. You can change these attributes here. The attributes you can change depend on the user interface element you selected. You can immediately see any changes made in the preview.

- Message area

In this area, potential conflicts in the configuration, such as tabs with the same name, are immediately displayed.

## Configuration Editor for Floorplan Manager



### Structure of the configuration editor

Every user interface element is defined and configured using its attributes. Your authorization profiles determine whether you can carry out a configuration or enhancement of user interface elements. The following two authorization profiles are of importance:

- S\_DEVELOP

With the authorization profile for ABAP Workbench, you can make any and all changes to a user interface developed with Web Dynpro ABAP.

- S\_WDR\_P13N

You can use this authorization profile to make changes to a user interface if the S\_DEVELOP authorization profile is not assigned to your user. It authorizes you to configure a Web Dynpro application in administrator mode.

For more information on these authorization profiles, see SAP Library for Web Dynpro ABAP under [Authorization Checks in Configuration/Personalization](#).

You can use the configuration editor to perform Web Dynpro built-in configuration as well as component-defined configurations of user interface elements. For more information on Web Dynpro built-in and component-defined configuration, see [Fitting Web Dynpro to Your Needs](#).

### Activities

You can launch the configuration editor for Floorplan Manager in one of the following ways:

- At runtime in [administrator mode](#), from the application user interface using the *Adapt Configuration* link.
- At runtime in [expert mode](#), from the application user interface using the *Change Configuration* link.
- At design time in *Web Dynpro Application Configuration* in the *Object Navigator* of the ABAP Workbench.
- At design time in *Web Dynpro Component Configuration* in the *Object Navigator* of the ABAP Workbench.



## Form Editor for Floorplan Manager

You use the form editor to adjust a form in an application to your specific business requirements. This is done by configuring [form components](#).

### Features

The form editor consists of the following work areas:

- Preview

In the preview, all form elements from the current configuration are displayed so as to give you a picture of the layout of the form.
- Hierarchy

All form elements (groups, melting groups, and elements) are displayed in the hierarchy.
- Attribute view

Attributes of the currently selected form element that can be changed using the form editor are displayed in the attribute view.
- Action area

The action area contains links to all the actions you can execute for the form component. The actions that can be selected depend on the concrete configuration of the form. This means that it can differ within a configuration.

The form editor provides you with the following actions:

- Add Group
- Add Melting Group
- Edit Feeder Class

- Edit Parameters
- Configure Toolbar
- Configure Group

The form editor provides you with the following functions for editing a group:

- Change Group Attributes

The group name, group type, and index can be changed.

- Add New Group
- Add Melting Group
- Add Element

You can select a field from the field catalog and determine the label text and display type.

- Delete Group

The form editor provides you with the following functions for editing a melting group:

- Add Group Element

You can select a field from the field catalog. Fields are configured in more detail by changing the group element attributes.

- Change Group Element Attributes

The display type, visibility of the label, label text, and index can be changed. Any other group element attributes that can be changed depend on the display type.

- Delete Group Element

The form editor provides you with the following functions for editing a toolbar:

- Add Button
- Change Button
- Delete Button

The form editor is launched in a separate browser window. You can launch the form editor in change or display mode and save your changes at any time.



Note

The component-defined processing view is pre-set. Make sure that this view is selected before configuring a form component.

The form editor launches from the [configuration editor for Floorplan Manager](#) automatically when you launch the configuration of an application-specific view (UIBB) that uses the `FPM_FORM_UIBB` Web Dynpro component.



## List Editor for Floorplan Manager

You use the list editor to adjust a list within an application to your specific business requirements. This is done by configuring [list components](#).

### Features

The list editor consists of the following work areas:

- Preview

In the preview, the list in the current configuration is displayed so as to give you a picture of the layout of the list.

- Hierarchy

All list elements (columns, toolbar, and parameters) are displayed in the hierarchy as a tree structure.

- Attribute view

Attributes of the currently selected list element that can be changed using the list editor are displayed in the attribute view.

- Action area

The action area contains links to all the actions you can execute for the list component. Which actions can be selected depends on the concrete configuration of the list. This means that the selection of actions can differ within a configuration.

The list editor provides you with the following actions:

- Edit Feeder Class
- Edit Parameters
- Configure Column
- Configure Toolbar

The form editor provides you with the following functions for editing a column:

- Add Column

You can select a field from the field catalog and determine the column header and display type.

- Delete Column

The list editor is launched in a separate browser window. You can launch the list editor in change or display mode and save your changes at any time.



Note

The component-defined processing view is pre-set. Make sure that this view is selected before configuring a list component.

The list editor launches from the [configuration editor for Floorplan Manager](#) automatically when you launch the configuration of an application-specific view (UIBB) that uses the `FPM_LIST_UIBB` Web Dynpro component.



## Tabbed Component Editor for Floorplan Manager

You use this editor to adjust a [tabbed component](#) within an application to your specific business requirements. This is done by configuring the component.

### Features

The editor consists of the following work areas:

- Preview

In the preview, all application-specific views (UIBBs) from the current configuration are displayed so as to give you a picture of the layout of the tabbed component.

- Layout

In this area, you determine whether the tabbed component should be arranged horizontally or vertically.

- Hierarchy

All application-specific views (UIBBs) are displayed in the hierarchy as a tree structure.

- Attribute view

Attributes of the currently selected application-specific view (UIBB) that can be changed using the editor are displayed in the attribute view.

- Action area

The action area contains links to all the actions you can execute for the tabbed component.

The editor for a tabbed component provides you with the following actions:

- Add Master Component (technical name: MASTER UIBB)

- Add Tab (technical name: TAB)
- Add Application-Specific View to Tab (technical name: TAB UIBB)

The editor for a tabbed component is launched in a separate browser window. You can launch the editor in change or display mode and save your changes at any time.



Note

The component-defined processing view is pre-set. Make sure that this view is selected before configuring a tabbed component.

The editor for a tabbed component launches from the [configuration editor for Floorplan Manager](#) automatically when you launch the configuration of an application-specific view (UIBB) that has the `FPM_TABBED_UIBB` Web Dynpro component.



## Design Time with the FPM Configuration Editor

Within the *FPM Configuration Editor*, the *Component Configuration* windows (for your OIF, GAF, and IDR components) help you to design the layout of your application.

Within the component configuration windows, you can perform the following tasks:

- Add extra steps or views (depending on your floorplan instance), including substeps and sub views
- Configure the toolbar with predefined buttons and navigation menus and attach events to these elements
- Attach your UIBBs to the relevant steps or views (or attach the FPM predefined [GUIBBs](#))
- Configure Quick Help for your application
- Configure an initial screen, a confirmation screen and extra variants for your application
- Change the global settings for your application and set variant parameters

The interface view of your application is the smallest unit of application UI that can be configured in the FPM. By assigning the interface view as a UIBB you are in effect composing how your application content area will look when the application runs within FPM.



## Floorplan Instances in the FPM Configuration Editor

What you see in the hierarchy of the *FPM Configuration Editor* depends on the type of Floorplan instance you are using in your application.

#### **OIF Instance**

In an OIF application, FPM displays your UIBBs in multiple tabs within the *Content Area*. The hierarchy displays the following types of views:

- *Main View*: These represent a single tab within the *Content Area* of your application. Attributes allow you to name and identify the individual tabs. Each Main View contains one or more sub-views.

To add more tabs, choose *Add Main View*.

- *Sub-View*: You add your UIBBs to the sub views. An FPM application must have at least one UIBB for each sub view. The *FPM Configuration Editor* automatically provides this, but you can add your own predefined UIBBs from your application. These UIBBs will be rendered one beneath the other. As well as containing UIBBs, sub views enable you to further divide your tabs for more complex applications. You can configure headings for both main- and sub-views. However, if you create only one main view with only one subview, then no tabs are displayed at all.

Adding UIBBs to a Sub View:

1. In the hierarchy, select the sub view in which you want to add your UIBB(s) and choose *Add UIBB*.
2. In the attribute view, enter the following details:
  - Component (the name of the Web Dynpro component implementing the FPM interface `IF_FPM_UI_BUILDING_BLOCK`)
  - View (the name of the above Web Dynpro component's Window containing the UIBB)

#### **GAF Instance**

In a GAF application, FPM displays your UIBBs as individual steps in the overall Roadmap. For GAF applications, the hierarchy displays the following types of steps:

- *Main Step*: Each main step in the hierarchy represents one roadmap step. An FPM application must have at least one UIBB for each main step. The *FPM Configuration Editor* automatically provides this but you can add your own predefined UIBBs from your application. Attributes allow you to name and identify the individual main steps.
- *Sub Step*: A substep is a step that appears between two main steps. Attributes allow you to name and identify the individual substeps. Like a main step, substeps must have at least one UIBB. You add UIBBS to a substep in the same way you add them to a sub view.

Substeps are not visible at startup, but all main steps that are a possible starting point for substeps are indicated as such on the Roadmap Element at runtime. Whether a substep is completed or not at runtime, depends on the



application context and the user input. Therefore, substeps are statically declared but activated at runtime by the application (via the FPM API).

For more information on adding substeps and dynamically activating them, see [Adding and Activating Substeps](#).



## Adding and Activating Sub-Steps

### Procedure

The configuration of substeps is similar to that of main steps. You can add one or more substeps to a main step and each substep can contain one or more UIBBs.

To add substeps, perform the following steps:

1. In the *FPM Configuration Editor*, open the *Component Configuration* window.
2. In the hierarchy, select the main step to which you want to add a substep. Choose *Add Sub-Step* from the action region.
3. In the attribute view, enter the following data:
  - **SUBSTEP**: Enter the *Component ID* and *Name*. The name will be shown at runtime beneath the corresponding substep.
  - **SUBSTEP\_UIBB**: Enter the *Component* and *Window*. In *Window* enter the name of the Web Dynpro window of the interface view (not the name of the Web Dynpro view itself).

After a substep has been configured statically, you may invoke it at runtime via the FPM API. This is done by raising a special FPM event. Before raising this event, the event parameters are populated with the corresponding substep ID that you want to use. This is shown in the sample code below:



### Syntax

```
1. DATA: lo_fpm TYPE REF TO if_fpm,
2.         lr_event TYPE REF TO cl_fpm_event.
3. * get reference to FPM API
4. lo_fpm = cl_fpm_factory=>get_instance( ).
5. * create event
6. lr_event = cl_fpm_event=>create_by_id(
   cl_fpm_event=>gc_event_change_step ).
7. * fill event parameters
8. lr_event->mo_event_data-set_value(
9.   iv_key   = cl_fpm_event=>gc_event_param_mainstep_id
10.  iv_value = <ID of Main Step> ).
11. lr_event->mo_event_data->set_value(
12.  iv_key = cl_fpm_event=>gc_event_param_substep_id
13.  iv_value = <ID of Sub-Step> ).
14. lr_event->mo_event_data->set_value(
15.  iv_key = cl_fpm_event=>gc_event_param_subvariant_id
16.  iv_value = <ID of Sub-Step variant> ).
17. * now raise event
18. Web_Dynpro_this->fpm->raise_event( io_event = lr_event )
```



## FPM Toolbar

FPM allows you to construct toolbars according to the SAP UI Guidelines. You choose which toolbar elements you require and FPM positions them in a predetermined location. FPM allows you to configure the following toolbar elements:

- Standard function buttons: buttons such as *Save*, *Edit*, *Finish*, *Read-Only*
- Application-specific buttons: buttons to which you add your own code
- Button choices: buttons which offer the user a dropdown menu with a list of further options. You can define the individual menu options in a button-choice and attach events to them. FPM provides no predefined events for these menu options but allows you to attach your own events instead. To attach your own predefined event to a button, enter a menu option name (*Label*) and the event ID. When the menu option is selected during run-time, the FPM will call up the attached event. A button choice is indicated in the *Add Toolbar Element* dialog box by a small arrow in the bottom right-hand corner of the button.
- Navigation menus



### Note

The *Close* button appears automatically on the FPM toolbar but you cannot configure it like the above standard function buttons. You can hide it by using the CNR API or with an application parameter `FPM_HIDE_CLOSE=X`.

## Differences between an OIF and a GAF Toolbar

### OIF Application

There is only one toolbar in every OIF variant. This toolbar contains more standard buttons than the GAF toolbar. Additionally, the OIF toolbar has two more options to create application-specific buttons. FPM automatically adds a *Save* button to an OIF toolbar when you create the component configuration. As the *Save* button belongs to the category Activation Function, you can configure it (e.g. with a tooltip, label or event).

### GAF Application

In a GAF application, every main step and substep inside a variant has its own toolbar. This enables you to have a different toolbar configuration at each step in the roadmap. FPM automatically adds the *Next* and *Previous* buttons to a GAF toolbar when you create the component configuration.



### Note

There is no ‘main’ toolbar in a GAF application. If you require a particular button on the toolbar at each step in the roadmap, you add it to each substep toolbar.

## Activities

### Adding Elements to a Toolbar

1. In the *FPM Configuration Editor*, locate the OIF or GAF component of your application and choose *Change*. This opens the OIF or GAF component configuration in edit mode.
2. To add an element to a toolbar, choose *Add Toolbar Element* in the action area. The *Add Toolbar Element* dialog box appears.
3. Select a button and choose *OK*. The button now appears in the hierarchy under *Toolbar* and the button’s editable attributes are visible in the preview.

### Adjusting the Toolbar Dynamically

During runtime the content and visibility of the OIF and GAF toolbars may be changed via the [Context Navigation Region \(CNR\) APIs](#). Note that there are different APIs for each floorplan type.

With these APIs you can dynamically change the FPM toolbars of both the initial screen the and the main screen.



### Toolbar Buttons

The following table describes some of the non self-explanatory toolbar buttons.

<i>Toolbar Button Name</i>	<i>Button Description</i>
<i>Activation Function</i>	This button is intended primarily to be used as a <i>Save</i> button. As most applications require a <i>Save</i> button, the <i>FPM Configuration Editor</i> automatically adds this button to your configuration by default. The FPM Event <code>FPM_Save</code> is set as the default FPM Event ID but you can edit this.
<i>Alternate Function</i>	Use this button when you need to call an application-specific function from your application screen. You can add your own application-specific event to it. This button appears in the same toolbar region as the <i>Activation (Save)</i> button.
<i>Other Function</i>	Use this button when you need to call an application-specific function from your application screen. You can add your own application-specific event to it. This button appears in a toolbar region separated from the <i>Activation (Save)</i> button. Additionally, it has attributes for Explanation and Button Design.

<i>You can Also/ Related Links</i> (navigation links)	These two toolbar elements provide navigation links away from the FPM. These elements require a Role and an Instance, both of which are taken from a launchpad which you must first create and configure.
<i>Close</i>	FPM provides GAF applications with a <i>Close</i> button. You cannot configure this button (when you select it at design time, you will see no attributes). For technical reasons, this button is not visible in every system (see CSN Note #1234843). If you need to hide this button – e.g. your application is executed within an iView on a portal page, please refer to chapters Using Application Parameters and Adjusting the toolbar using the CNR API.
<i>Exit to Main Step</i> (GAF only)	This is available only to sub-steps. If you click this button during run-time you return to the Main Step to which the button is assigned.
<i>Finish</i>	This is available only to main steps. If you click this button during run-time, the roadmap is executed sequentially; the FPM will navigate automatically through the roadmap as far as the last screen (before the confirmation screen) or will stop prematurely if it encounters an error.
<i>Next Step/Final Action</i> (GAF only)	Extra attributes are available for <i>Next Step</i> in the final roadmap step. These attributes (Label, Event ID) allow you to execute your own predefined final action before the confirmation screen. Each variant of the roadmap can have one <i>Final Action</i> .

#### Toolbar Element Attributes

Toolbar elements have a variety of attributes and not every element has the same attributes. The table lists some of the non self-explanatory toolbar button attributes.

<b>Toolbar Element Attribute</b>	<b>Attribute Description</b>
<i>Element ID</i>	Enter an Element ID if you want to change the properties of a toolbar element dynamically during runtime.
<i>Duplicate Toolbar</i>	This allows you to display a copy of the toolbar at the bottom of your application screen.
<i>Sequence Index</i>	This allows you to choose the order in which your application-specific UI elements (e.g. Other Function buttons, Main Steps) appear on the toolbar or in the hierarchy. The toolbar elements which FPM automatically adds to the toolbar can not be rearranged using this attribute.
<i>Repeat Sel.</i>	This is available for button-choice elements. If you tick this

<b>Toolbar Element Attribute</b>	<b>Attribute Description</b>
<i>Action</i> (Repeat Select Action)	checkbox, the menu option that is selected from a button-choice at run-time will then be visible as the button choice title for the current session. If the user wishes to select the same option next time, he must click only the button and not scroll through the list of menu options.
<i>Enabled</i>	This greys out a toolbar element; it renders a toolbar element unusable if the checkbox is not ticked.
<i>Visibility</i>	If you check the visibility attribute of both the button and the button-choice, only the button is visible in the toolbar.

### Toolbar Button Events

Every Standard Function button is attached to an FPM event (for example, *Edit* is connected to the FPM event `gc_event_edit`). The connection to these raised FPM events is hard-coded and cannot be changed. The event can, of course, be changed dynamically by calling other events. Some button events are pre-configured by the FPM (for example, the *Previous* and *Next* navigation button events and the *Save* button event) and require no extra code, but generally the application must provide the event processing. In general, the FPM ensures only that all affected UIBBs are informed. For example, although the FPM provides a *Print* button, there is no print support in FPM. FPM provides this button only to ensure that it is rendered according to the SAP UI Guidelines. The application must provide the necessary print functions.

The table below lists the toolbar buttons (and button-choices) and the events that they raise.

<b>Toolbar Button</b>	<b>Event Raised</b>	<b>Floorplan Instance</b>
<i>Activation Function</i>	self-defined via configuration	OIF
<i>Alternate Activation</i>	self-defined via configuration	OIF
<i>Check</i>	<code>gc_event_check</code>	OIF
<i>Close</i>	<code>gc_event_close</code>	OIF and GAF
<i>Delete Object</i>	<code>gc_event_delete_current_object</code>	OIF
<i>Edit</i>	<code>gc_event_edit</code>	OIF

<b>Toolbar Button</b>	<b>Event Raised</b>	<b>Floorplan Instance</b>
<i>Exit to Main Step</i>	gc_event_exit_to_main_step	GAF
<i>Load Draft</i>	gc_event_load_draft	OIF
<i>New</i>	gc_event_new	OIF
<i>Next Object</i>	gc_event_next_object	OIF
<i>Next Step</i>	If no final action is defined: gc_event_next_step If a final action is defined: the self-configured event in the final action node and the next step event are raised	GAF
<i>Other function</i>	self-defined via configuration	OIF and GAF
<i>Previous Object</i>	gc_event_previous_object	OIF
<i>Previous Step</i>	gc_event_previous_step	GAF
<i>Print</i>	gc_event_print	OIF
<i>Print Preview</i>	gc_event_print_preview	OIF
<i>Read Only</i>	gc_event_read_only	OIF
<i>Redo</i>	gc_event_redo	OIF
<i>Refresh</i>	gc_event_refresh	OIF
<i>Save As</i>	gc_event_save_as	OIF
<i>Save Draft</i>	gc_event_save_draft	OIF and GAF
<i>Send</i>	gc_event_send	OIF
<i>Start Over</i>	gc_event_start_over	OIF
<i>Undo</i>	gc_event_undo	OIF
<b>Toolbar Button-Choice</b>	<b>Event Raised</b>	<b>Floorplan Instance</b>

<b>Toolbar Button-Choice</b>	<b>Event Raised</b>	<b>Floorplan Instance</b>
<i>Send</i>	self-defined via configuration	OIF
<i>Print</i>	self-defined via configuration	OIF
<i>Print Preview</i>	self-defined via configuration	OIF
<i>New</i>	self-defined via configuration	OIF



## **IF\_FPM\_CNR\_GAF Interface**

This interface provides you with methods to dynamically change the FPM toolbar of an initial screen or main screen.

The interface is accessed via the `CL_FPM_SERVICE_MANAGER`, as the code below shows:

### **Accessing the API for a GAF application:**



#### **Syntax**

```

1. DATA: lo_cnr_gaf TYPE REF TO if_fpm_cnr_gaf,
2.         lo_fpm TYPE REF TO if_fpm.
3. lo_fpm = cl_fpm_factory=>get_instance( ).
4. lo_cnr_gaf ?= lo_fpm->get_service(
   cl_fpm_service_manager=>gc_key_cnr_gaf ).

```

### **Methods**

This interface provides you with the methods described in the table below.

<b>Method Name</b>	<b>Method Description</b>
DEFINE_BUTTON	With this method either standard buttons or application-specific buttons can be created and edited. The parameter <code>IV_FUNCTION</code> defines the button type (see <code>IF_FPM_CONSTANTS=&gt;gc_button</code> ). The <code>ELEMENT_ID</code> is needed if application-specific buttons must be changed subsequently.
DEFINE_BUTTON_CHOICE	With this method either standard button-choices or application-specific button-choices can be created and edited. The parameter <code>IV_FUNCTION</code> defines the button-choice type (see <code>IF_FPM_CONSTANTS=&gt;gc_button_choice</code> ). The <code>ELEMENT_ID</code> is needed if application-specific buttons must be changed subsequently.

<b>Method Name</b>	<b>Method Description</b>
CREATE_SEPARATOR	Use this method to create a separator at runtime in the OTHER_FUNCTIONS area (application-specific).
SET_DUPLICATE_TOOLBAR	Use this method to activate or deactivate the duplication of the toolbar.
DEFINE_YOU_CAN_ALSO	Use this method to define launchpads for the <i>You Can Also</i> menu bar for (see Navigation API chapter).
DEFINE_RELATED_LINKS	Use this method to edit the menu bar for RELATED_LINKS (see Navigation API chapter).
GET_BUTTONS	This method determines which buttons (and their configurations) are to be shown in the toolbar.
GET_BUTTON_CHOICES	This method determines which button-choices (and their configurations) are to be shown in the toolbar.
GET_SEPARATORS	This method determines the positions of the separators in the toolbar (only in the <i>Other Functions</i> area).
GET_RELATED_LINKS	This method determines the contents of the <i>Related Links</i> menu in the toolbar.
GET_YOU_CAN_ALSO	This method determines the contents of the <i>You Can Also</i> menu in the toolbar.

#### **GAF Specific Parameters**

Depending on the location of the UI elements that you wish to define, the following parameters (outlined in the table below) are passed with every GAF CNR API method:

<b>Location of UI Elements</b>	<b>Parameters</b>
Main Step	<ul style="list-style-type: none"> <li>• VARIANT_ID</li> <li>• MAINSTEP_ID</li> </ul>
Sub-Step	<ul style="list-style-type: none"> <li>• VARIANT_ID</li> <li>• MAINSTEP_ID</li> <li>• SUBVARIANT_ID</li> <li>• SUBSTEP_ID</li> </ul>
Initial Screen	<ul style="list-style-type: none"> <li>• Screen</li> </ul>

#### **Example**



An example of method calls to change the CNR of the GAF at runtime is shown below:



### Syntax

```

1.  DATA: lo_cnr_gaf TYPE REF TO if_fpm_cnr_gaf,
2.          lo_fpm TYPE REF TO if_fpm.
3.  lo_fpm = cl_fpm_factory=>get_instance( ).
4.  lo_cnr_gaf ?= lo_fpm->get_service(
   cl_fpm_service_manager=>gc_key_cnr_gaf ).
5.  lo_cnr_gaf ->define_button(
6.      EXPORTING
7.          iv_variant_id      = < optional; e.g. 'variant_1'; current
   variant if skipped >
8.          iv_mainstep_id     = < optional; 'mainstep_1'; current
   mainstep if skipped >
9.          iv_subvariant_id   = < optional; 'subvariant_xyz' >
10.         iv_substep_id      = < optional; 'substep_99' >
11.         iv_function        = < e.g. EXIT_TO, FINISH,
   OTHER_FUNCTIONS (appl-specific buttons), SAVE_DRAFT, NEXT_STEP)
   see also IF_FPM_CONSTANTS=>gc_button >
12.         iv_screen          = < optional; the screen where the UI-
   Element has to be changed (INIT, MAIN) >
13.
14.         iv_element_id      = < optional; only if you want to
   change the properties of application-specific buttons
   afterwards>
15.         iv_sequence_id     = < optional; only if you use
   OTHER_FUNCTIONS; determines the place where to insert this
   button >
16.         iv_design          = < optional; Button-Design >
17.         iv_enabled         = < optional; Button-Enabling >
18.         iv_explanation      = < optional; Button-Explanation >
19.         iv_on_action       = < optional; determines the Event-Id
   for a button; not possible with standard buttons >
20.         iv_text            = < optional; Button-Label >
21.         iv_tooltip         = < optional; Button-Tooltip >
22.         iv_visibility      = < optional; Button-Visibility >
23.         iv_default_button  = < optional; only for NEXT button; by
   pressing enter within an application triggers the action of
   this button> ).
24.
25.         iv_hotkey          = < optional; key-combination for
   activating the event of this button>
26.
27.  lo_cnr_gaf->define_button_choice(
28.      EXPORTING
29.          iv_variant_id      = < optional; e.g.'variant_1';
   current variant if skipped >
30.
31.          iv_mainstep_id     = < optional; 'mainstep_1'; current
   mainstep if skipped >
32.          iv_subvariant_id   = < optional; 'subvariant_xyz' >
33.          iv_substep_id      = < optional; 'substep_99' >
34.          iv_function        = < e.g. OTHER_FUNCTIONS (appl-
   specific button-choices)>
35.          iv_screen          = < optional; the screen where the
   UI-Element has to be changed(INIT, MAIN) >
36.

```

```

37.     iv_element_id      = < optional; only if you want to
      change the button-choice properties afterwards>
38.     iv_sequence_id    = < optional; only if you use
      OTHER_FUNCTIONS; determines the place where to insert this
      button-choice >
39.     iv_enabled        = < optional; Button-Choice-Enabling
      >
40.     iv_text           = < optional; Button-Choice-Label >
41.     iv_tooltip        = < optional; Button-Choice-Tooltip
      >
42.     iv_visibility     = < optional; Button-Visibility >
43.     it_menu_action_items = < menu elements of a Button-Choice
      > ).

```



## IF\_FPM\_CNR\_OIF Interface

This interface provides you with methods to dynamically change the FPM toolbar of an initial screen or main screen.

The interface is accessed via the `CL_FPM_SERVICE_MANAGER`, as the code below shows:

### Accessing the API for an OIF application:



#### Syntax

```

1.  DATA: lo_cnr_oif TYPE REF TO if_fpm_cnr_oif,
2.         lo_fpm TYPE REF TO if_fpm.
3.  lo_fpm = cl_fpm_factory=>get_instance( ).
4.  lo_cnr_oif ?= lo_fpm-
      >get_service( cl_fpm_service_manager=>gc_key_cnr_oif ).

```

### Methods

This interface provides you with the methods described in the table below.

Method Name	Method Description
DEFINE_BUTTON	With this method either standard buttons or application-specific buttons can be created and edited. The parameter <code>IV_FUNCTION</code> defines the button type (see <code>IF_FPM_CONSTANTS=&gt;gc_button</code> ). The <code>ELEMENT_ID</code> is needed if application-specific buttons must be changed subsequently.
DEFINE_BUTTON_CHOICE	With this method either standard button-choices or application-specific button-choices can be created and edited. The parameter <code>IV_FUNCTION</code> defines the button-choice type (see <code>IF_FPM_CONSTANTS=&gt;gc_button_choice</code> ). The <code>ELEMENT_ID</code> is needed if application-specific buttons must

Method Name	Method Description
	be changed subsequently.
CREATE_SEPARATOR	Use this method to create a separator at runtime in the OTHER_FUNCTIONS area (application-specific).
SET_DUPLICATE_TOOLBAR	Use this method to activate or deactivate the duplication of the toolbar.
DEFINE_YOU_CAN_ALSO	Use this method to define launchpads for the <i>You Can Also</i> menu bar for (see Navigation API chapter).
DEFINE_RELATED_LINKS	Use this method to edit the menu bar for RELATED_LINKS (see Navigation API chapter).
GET_BUTTONS	This method determines which buttons (and their configurations) are to be shown in the toolbar.
GET_BUTTON_CHOICES	This method determines which button-choices (and their configurations) are to be shown in the toolbar.
GET_SEPARATORS	This method determines the positions of the separators in the toolbar (only in the <i>Other Functions</i> area).
GET_RELATED_LINKS	This method determines the contents of the <i>Related Links</i> menu in the toolbar.
GET_YOU_CAN_ALSO	This method determines the contents of the <i>You Can Also</i> menu in the toolbar.

### OIF Specific Parameters

Since a toolbar exists for every OIF variant, only the `VARIANT_ID` must be passed with every OIF CNR API method.

### Example

An example of method calls to change the CNR of the OIF at runtime is shown below:



#### Syntax

```

1. DATA: lo_cnr_oif TYPE REF TO if_fpm_cnr_oif,
2.         lo_fpm TYPE REF TO if_fpm.
3. lo_fpm = cl_fpm_factory=>get_instance( ).
4. lo_cnr_oif ?= lo_fpm->get_service(
5.   cl_fpm_service_manager=>gc_key_cnr_oif ).
6. lo_cnr_oif->define_button(
7.   EXPORTING
8.     iv_variant_id      = < optional; e.g. 'variant_1'; current
   variant if skipped >
9.     iv_function        = < e.g. ACTIVATION_FUNCTIONS (appl-
   specific buttons),ALTERNATE_FUNCTIONS (appl-specific buttons),

```

CHECK, DELETE\_OBJECT, EDIT, LOAD\_DRAFT, NEW, NEXT\_OBJECT, OTHER\_FUNCTIONS (appl-specific buttons), REVIOUS\_OBJECT, PRINT, PRINT\_PREVIEW, READ\_ONLY,REDO, REFRESH, SAVE\_AS, SAVE\_DRAFT, SEND, START\_OVER, UNDO, see also IF\_FPM\_CONSTANTS=>gc\_button >

- 9.
10. iv\_screen = < optional; the screen where the UI-Element has to be changed (INIT, MAIN) >
- 11.
12. iv\_element\_id = < optional; only if you want to change the button properties afterwards >
13. iv\_sequence\_id = < optional; only if you use OTHER\_FUNCTIONS; determines the place where to insert this button >
14. iv\_design = < optional; Button-Design >
15. iv\_enabled = < optional; Button-Enabling >
16. iv\_explanation = < optional; Button-Explanation >
17. iv\_on\_action = < optional; determines the Event-Id for a button; not possible with standard buttons >
18. iv\_text = < optional; Button-Label >
19. iv\_tooltip = < optional; Button-Tooltip >
20. iv\_visibility = < optional; Button-Visibility >
21. iv\_default\_button = < optional; only for buttons CHECK and REFRESH; by pressing enter within an application triggers the action of this button >
22. iv\_hotkey = < optional; key-combination for activating the event of this
23. button >
24. lo\_cnr\_oif->define\_button\_choice(  
25. EXPORTING  
26. iv\_variant\_id = < optional; e.g. 'variant\_1'; current variant if skipped >  
27. iv\_function = < e.g. NEW, OTHER\_FUNCTIONS (appl-specific button-choices), PRINT, PRINT\_PREVIEW, SEND, see also IF\_FPM\_CONSTANTS=>gc\_button\_choice >  
28. iv\_screen = < optional>; the screen where the UI-Element has to be changed (INIT, MAIN) >  
29. iv\_element\_id = < optional; only if you want to change the button-choice properties afterwards >  
30. iv\_sequence\_id = < optional; only if you use OTHER\_FUNCTIONS; determines the place where to insert this button-choice >  
31. iv\_enabled = < optional; Button-Choice-Enabling >  
32. iv\_text = < optional; Button-Choice-Label >  
33. iv\_tooltip = < optional; Button-Choice-Tooltip >  
34. iv\_visibility = < optional; Button-Visibility >  
35. it\_menu\_action\_items = < menu elements of a Button-Choice >  
>



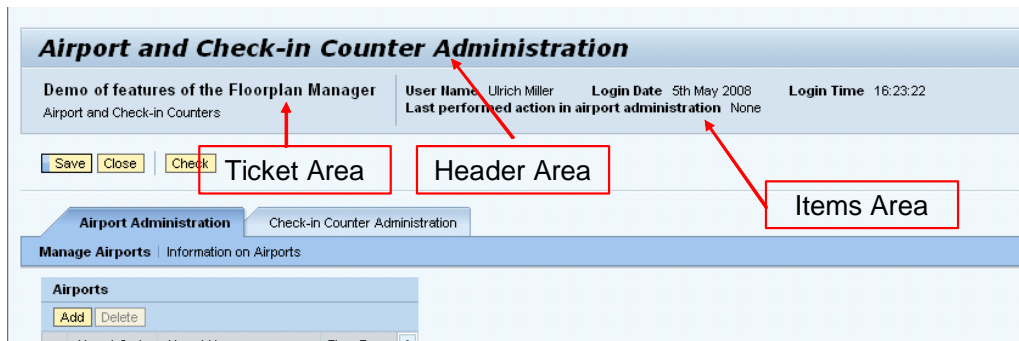
## FPM Identification Region (IDR)

The *Identification Region* (IDR) consists of the following three areas:

- Header area (IDR Basic)
- Ticket area (IDR Extended)

- Items area

This is illustrated in the figure below:



Both the header and the ticket areas can be configured at design-time in the *Component Configuration* window for the IDR configuration.

Note the following points regarding the ticket area:

- The ticket area is only available for OIF applications.
- To configure the ticket area, choose *Add IDR Extended*.

Attributes for *Ticket Top* and *Ticket Bottom* appear. These attributes can be called dynamically to add label/value pairs, label/navigation link pairs and label/icon pairs to the ticket area.

#### Adjusting the IDR Dynamically

During runtime, use the IDR API to make changes to the individual IDR areas. This API consists of the methods encapsulated in the [IF\\_FPM\\_IDR interface](#).

#### Adding a Link to the FPM Configuration Editor in the IDR

You can provide your application with a link to the *FPM Configuration Editor* from the IDR. For more information, see [Providing a Link to the FPM Configuration Editor](#).



This interface provides you with methods to change the IDR dynamically at run-time.

The sample code below shows you how to access this interface:



#### Syntax

```

1. DATA: lo_idr TYPE REF TO if_fpm_idr,
2.         lo_fpm TYPE REF TO if_fpm.
3. lo_fpm = cl_fpm_factory=>get_instance( ).
4. lo_idr ?= lo_fpm->get_service(
    cl_fpm_service_manager=>gc_key_idr ).

```

There are methods available for each of the following IDR areas:

- Header Area
- Ticket Area
- Items Area

#### Methods for IDR Header Area

Method Name	Method Description
GET_APPLICATION_TITLE	Retrieves the title text and its tooltip.
SET_APPLICATION_TITLE	Displays a new title text and tooltip in the header area.
SET_HEADER_VISIBILITY	Makes the header area visible or invisible.

#### Methods for IDR Ticket Area

Method Name	Method Description
GET_TICKET	Retrieves the texts of the ticket top, ticket bottom and their tooltips.
SET_TICKET	Displays new texts of the ticket top, ticket bottom and their tooltips.
SET_TICKET_VISIBILITY	Makes the ticket area visible or invisible.

#### Methods for Items Area

Method Name	Method Description
ADD_ITEM_GROUP_BY_VAL	Adds a new item group to the item area. One item consists of a label, its tooltip, a value and the value's tooltip. A group of items consists of an arbitrary amount of such items. With this method, you can add items to the IDR as simple static text strings. Therefore, if the value of an item needs to be changed

Method Name	Method Description
	<p>at a later point in time, you will need to call method <code>CHANGE_ITEM_GROUP_BY_VAL</code>. The method <code>ADD_ITEM_GROUP_BY_REF</code> can also be used to pass references to Web Dynpro context nodes to the IDR. In this case, the value changes automatically when the value of the corresponding attribute in the context node changes.</p>
<code>CHANGE_ITEM_GROUP_BY_VAL</code>	<p>Changes the label and values that were passed to the IDR via the method <code>ADD_ITEM_GROUP_BY_VAL</code>.</p>
<code>ADD_ITEM_GROUP_BY_REF</code>	<p>Similar to <code>add_item_group_by_val</code>. Adds label/value items to the IDR. In this case, the value is not passed as a static text but as reference to a Web Dynpro context node attribute. The advantage here is that the value can be of a type other than string. In addition, updating the value happens automatically; whenever the attribute of the context node changes, the IDR changes the visible value. It is also possible for the IDR to show the unit of the value. Do this using a flag; the actual unit is taken from the DDIC information of the value's type. Therefore, this feature will only work if the type of the attribute in the context node, (which is passed to the IDR) has a defined DDIC unit.</p>
<code>ADD_NAVIGATION_ITEM</code>	<p>Adds a pair of label/navigation links to the IDR. The link itself is provided by the report launchpad. It makes no difference whether the link in the report launchpad is supplied by the database or is created dynamically during runtime via the report launchpad API. For more information about the report launchpad, refer to the report launchpad documentation. You specify the launchpad via the parameters <code>instance</code> and <code>role</code>. Since one launchpad may contain several targets (and this method is used to add only one target), use an additional parameter to specify the single target. The additional parameter is either the application alias or the navigation key.</p>
<code>CHANGE_NAVIGATION_ITEM</code>	<p>Use this method to edit a pair of label / navigation links that you added using the method <code>ADD_NAVIGATION_ITEM</code>. It is possible to change only the label and the link text with this method. If you want to change the target itself, use the report launchpad API.</p>
<code>ADD_IMAGE_ITEM</code>	<p>Adds pairs of label/icons to the IDR.</p>

Method Name	Method Description
CHANGE_IMAGE_ITEM	Edits a label/icon pair that you added using the method ADD_IMAGE_ITEM.
CONTAINS_ITEM_GROUP	Checks whether a certain item group exists within the IDR.
REMOVE_ITEM_GROUP	Removes a certain item group from the IDR.
INITIALIZE_ITEMS	Clears all items from the IDR.
SET_ITEMS_VISIBILITY	Edits the visibility of the item area (i.e. the visibility status of all items, not just single items).



## Providing a Link to the FPM Configuration Editor in the IDR

There are currently two options to provide a link (in the IDR header area of your FPM application) which points to the *FPM Configuration Editor*:

- Using transaction SU3. To do this, proceed as follows:
  1. Open transaction SU3 and choose the *Parameters* tab.
  2. Add the parameter `FPM_CONFIG_EXPERT` and set the Parameter Value to X.

The *Change Configuration* link appears in the IDR header area when you start the *FPM Configuration Editor*, via the *Web Dynpro Explorer*, for your application configuration. This corresponds to a change of the explicit and implicit configuration in development mode.

- Starting your FPM application with URL parameter `sap-config-mode=X`.

The link *Adapt Configuration* appears in the IDR header area when you start the *FPM Configuration Editor* via Web Dynpro application `customize_component`. This corresponds to a customizing of the explicit and implicit configuration in the administrator mode. In the administrator mode you may adapt all elements of the configuration that have not been marked previously as final elements in the development mode.



## Quick Help

You can use this function in a floorplan to provide application users with a quick help that gives a helpful explanation of a subview, initial screen, main step, or substep. The quick help is only displayed if the user has activated it using the context menu.



## Features

You can either enter the quick help text directly or give a reference to a documentation object. It is a good idea to use a reference to a documentation object when the content of the quick help is used in multiple views or applications. If you enter a text directly and enter a reference to a documentation object, then the content of the documentation object is displayed as quick help.

You can display the quick help using the application's context menu. You can create, change, or delete quick help texts.



Note

The quick help text is stored in the hierarchical view of the configuration editor either in the *Text* or *Documentation Object* attribute of an *Explanation* hierarchy element.

When you create a quick help from the action area of the configuration editor of Floorplan Manager, the *Explanation* hierarchy element is created below the corresponding hierarchy element (for example, subview details). Additionally, a suggested text is created by the system in the *Text* attribute.

You can also delete a quick help completely by selecting the *Delete* function in the attribute view of an *Explanation* hierarchy element.



## Create Quick Help

### Procedure

To create a quick help in the configuration editor, you can either enter the quick help text directly or enter a reference to a documentation object.

#### Create Quick Help as Direct Text

1. Select a Web Dynpro application configuration in the *Object Navigator* of the *ABAP Workbench*.
2. In the *Web Dynpro Explorer: Display Web Dynpro Configuration* screen, choose ► *Web Dynpro Configuration* → *Test* → *Execute in Administrator Mode* ↩.

The Web Dynpro application is launched in a separate browser window.

3. In this window, go to the application's identification region and choose the *Adapt Configuration* link.
4. In the *Editor for Web Dynpro ABAP Components — Customizing* screen, choose *Change*.
5. On the *Component Customizing <application name>* screen, make sure that the *Component-Defined* view is on.
6. Choose *Expand All*.

7. Select the subview, main step, substep, or initial screen and choose *Add Explanation* in the action area of the configuration editor.



Note

The system automatically generates a suggested text for the explanation in the *Text* field. The suggested text is composed of the title of the subview, main step, substep, or initial screen and the prefix "explanation". For example, if a substep has the title *Details*, then the suggested text is *Explanation Details*.

8. In the *Text* field in the attribute view, overwrite the suggested text with the quick help text you would like.
9. Save the configuration.
10. Test the new configuration.

#### Create Quick Help Linking to a Documentation Object

1. To create a documentation object, choose ► *SAP Menu* → *Tools* → *ABAP Workbench* → *Utilities* → *SE61 – Documentation* ⏏.
  1. Choose *General Text* as the document class.
  2. Enter a technical name for the documentation object.
  3. Choose *Create*.
  4. Then enter desired quick help text.
  5. Choose *Save Active*.

The documentation object is now created and can be assigned as a quick help.
2. Select a Web Dynpro application configuration in the *Object Navigator* of the *ABAP Workbench*.
3. In the *Web Dynpro Explorer: Display Web Dynpro Configuration* screen, choose ► *Web Dynpro Configuration* → *Test* → *Execute in Administrator Mode* ⏏.

The Web Dynpro application is launched in a separate browser window.
4. In this window, go to the application's identification region and choose the *Adapt Configuration* link.
5. In the *Editor for Web Dynpro ABAP Components — Customizing* screen, choose *Change*.
6. On the *Component Customizing <application name>* screen, make sure that the *Component-Defined* view is on.

7. Select the subview, main step, substep, or initial screen for which you would like to add a quick help and choose *Add Explanation* in the action area of the configuration editor.



#### Note

The system automatically generates a suggested text for the explanation in the *Text* field.

8. In the *Documentation Object* field of the attribute view, enter the technical name of the documentation object.
9. Save the configuration.
10. Test the new configuration.



## Variants

In some cases the final configuration of an OIF view switch or a GAF roadmap may only be decided at runtime. For example, assume that an initial screen asks you to select one of three options. The subsequent roadmap or view switch that appears is dependent on the option you selected in the initial screen. FPM makes this possible by allowing you to configure variants. Each variant is a complete set of configuration data for an OIF view switch or a GAF roadmap. You use the input from the initial screen (or from other startup information, such as application parameters) to select one of the variants.

### Configuring Variant Selection

Variant selection is controlled programmatically with an application-specific configuration controller (AppCC).

To configure variant selection, proceed as follows:

1. Implement the interface `IF_FPM_OIF_CONF_EXIT` (or `IF_FPM_GAF_CONF_EXIT`) in one of the application components or in a new component. This interface has only one method `OVERRIDE_EVENT_OIF` (or `OVERRIDE_EVENT_GAF`) which passes a handler object of type `IF_OIF` (respective `IF_GAF`). This handler object provides the API with information to manipulate the floorplan configuration at runtime.
2. To select the variant, use the `SET_VARIANT` method of this object as follows:

#### *OIF Instance*



#### Syntax

```
1.     method OVERRIDE_EVENT_OIF .
2.         ...
3.         case io_oif->mo_event->MV_EVENT_ID.
```

```

4.         when if_fpm_constants=>gc_event-
      leave_initial_screen.
5.         io_oif->set_variant( ).
6.         ...

```

### *GAF Instance*



#### Syntax

```

7.     method OVERRIDE_EVENT_GAF .
8.         ...
9.         case io_gaf->mo_event->MV_EVENT_ID.
10.            when if_fpm_constants=>gc_event-
      leave_initial_screen.
11.            io_gaf->set_variant( ).

```



#### Note

In this sample code the variant selection takes place after the initial screen is exited. This is the latest point at which it is possible to select the variant. You can, however, select the variant at an earlier stage.

The last thing to do is to declare the AppCC to the FPM:

1. In the *FPM Configuration Editor*, open the component configuration editor window. In the control region, choose ► *Change* → *Global Settings* ⚡.
2. In the *Global Settings* dialog box, under *Application-Specific Parameters*, enter the Web Dynpro Component which you are using as an application-specific configuration controller.
3. Choose *Save*.



### **Initial Screen**

The *Initial Screen* is an optional screen. It is composed of one or more UIBBs.

### **Activities**

#### **Adding an Initial Screen**

1. Start the *FPM Configuration Editor* and open the OIF or GAF component configuration of your component.
2. In the control region, choose ► *Add* → *Initial Screen* ⚡.
3. Choose ► *Add UIBB* → *Add UIBB* ⚡ and enter the component (ID) and the relevant component window.

The FPM adds the *Start Button* automatically to the toolbar of the initial screen. It is non configurable. When you choose this button at run-time, FPM raises the event

IF\_FPM\_CONSTANTS=>GC\_EVENT-LEAVE\_INITIAL\_SCREEN, exits the initial screen, and displays the first roadmap step (in GAF instances) or View Switch (in OIF instances). Occasionally, you need to omit the initial screen from your application. If this is the case, raise the LEAVE\_INITIAL\_SCREEN event within your application-specific code:

#### Syntax

```
1. data: lo_fpm type ref to if_fpm
2. lo_fpm = cl_fpm_factory=>get_instance( )
3. lo_fpm->raise_event_by_id(IF_FPM_CONSTANTS=>GC_EVENT-
    LEAVE_INITIAL_SCREEN)
```

If your application has no initial screen, FPM displays the view switch (OIF) or the first roadmap step (GAF) at start-up.

#### Skipping the Initial Screen

OIF and GAF applications may start with an initial screen, in which you select the object you intend to work with. If the object is already known by the application (e.g. you are calling the application with the parameters already set), the initial screen is unnecessary. To skip an initial screen at runtime, proceed as follows:

Launch the FPM event LEAVE\_INITIAL\_SCREEN. You can launch this one of two ways:

- in the `OVERRIDE_EVENT_*`-method of your AppC
- in the `PROCESS_BEFORE_OUTPUT` method of one of your initial screen UIBBs (if you are not using an AppCC):

#### Syntax

```
1. data: lo_fpm type ref to if_fpm,
2. lv_object_id type string.
3. * Check event id
4. if lv_event_id = if_fpm_constants=>gc_event_start.
5. * Determine if Parameter OBJECT_ID is provided
6. lo_fpm = cl_fpm_factory=>get_instance( ).
7. lo_fpm->mo_app_parameter->get_value(
8. exporting iv_key = 'OBJECT_ID'
9. importing ev_value = lv_object_id ).
10. * In case OBJECT_ID is set, navigate directly to the
    main floorplan * area
11. if not lv_object_id is initial.
12. lo_fpm->raise_event_by_id(
13. if_fpm_constants=>gc_event-
    leave_initial_screen ).
14. endif
15. endif
```

#### Confirmation Screen

FPM does not provide a confirmation screen by default. However, a confirmation screen is available to both the OIF and GAF instances.

Confirmation screens may be variant dependent (that is, each variant of your application may require a different confirmation screen). FPM allows you to configure a confirmation screen for each variant.

### Confirmation Screen in OIF Instances

In OIF applications, the confirmation screen appears only when the object currently being processed in the application is deleted. After an object has been deleted, the confirmation screen appears in place of the normal view switch. Note that if your application does not include a delete function, you do not need a confirmation screen.

### Confirmation Screen in GAF Instances

Most GAF applications use a final confirmation step at the end of the roadmap. This confirmation step informs the user that the action he has just executed has been completed successfully. The configuration of the `FPM_GAF_COMPONENT` explicitly supports such a use case.

### Adding and Configuring the Confirmation Screen

To add a *Confirmation Screen* to your application, perform the following steps:

1. Start the *FPM Configuration Editor* of your application component and open the *Component Configuration* screen.
2. In the control region, choose **Add** → *Confirmation Screen* **←**. You can then select the radio buttons in the hierarchy to move between the *Confirmation Screen* and other screens in your application.

You display the confirmation screen in OIF applications by pressing the standard DELETE button during run-time. This raises the event `IF_FPM_CONSTANTS=>GC_EVENT-DELETE_CURRENT_OBJECT`. You can also raise this event within your application-specific code:



#### Syntax

```
1. data: lo_fpm type ref to if_fpm
2. lo_fpm = cl_fpm_factory=>get_instance( )
3. lo_fpm->raise_event_by_id(IF_FPM_CONSTANTS=>GC_EVENT-
  DELETE_CURRENT_OBJECT)
```



### FPM Event Loop

In Web Dynpro ABAP programming, a user interaction is reflected by a Web Dynpro action. If you require a user interaction to affect not only a local component but other components in the application too, the Web Dynpro action must be transferred to an FPM event.

This FPM event then passes through an FPM phase model (Event Loop) which is integrated into the Web Dynpro phase model. Within the FPM event loop all involved components can participate in the processing of the event.

If the FPM event results in another screen assembly (for example, navigation to another step in a GAF application or the selection of another view or sub view in an OIF application), the FPM handles this itself; there is no need for the application to fire plugs or similar.

## Activities

### Raising Standard Events

In a floorplan-based application, most events are triggered when a user chooses *Next* or *Previous* (in a GAF instance) or when switching from one view to another (in an OIF instance). For these interactions, the FPM automatically initiates the FPM event loop. Furthermore, these standard events are handled generically by the FPM.

However, there are scenarios where a standard event needs to be triggered from within an application-specific UIBB e.g. by-passing the initial screen if all necessary start-up parameters have been provided as URL parameters.

Each FPM event is represented at runtime by an instance of the class `CL_FPM_EVENT`. This class encapsulates all information (including the ID and additional, optional parameters) which is needed to execute the event.

### Triggering the FPM Event Loop

To trigger an FPM event loop, you complete the following two steps:

1. Create an instance of `CL_FPM_EVENT` with the appropriate attributes. For all the standard event IDs, there are constants available in the `IF_FPM_CONSTANTS` interface.
2. Raise the event by calling the method `IF_FPM~RAISE_EVENT` and passing on the instance of `CL_FPM_EVENT`.



#### Note

When an event requires no additional parameters, other than the event ID, the FPM offers an additional method `RAISE_EVENT_BY_ID`. This makes Step 1 above obsolete. In this case, raise the FPM event as detailed in the sample code below:



#### Syntax

- ```
1. data lo_fpm type ref to if_fpm
2. lo_fpm = cl_fpm_factory=>get_instance( )
3. lo_fpm->raise_event_by_id( IF_FPM_CONSTANTS=>GC_EVENT-
    LEAVE_INITIAL_SCREEN )
```

Since it is unknown whether the event can be executed successfully or not at the point the event is raised, do not enter code after the call to `RAISE_EVENT[_BY_ID]`.

## Triggering Application-Specific Events

To raise an application-specific event, follow the same rules as described in Triggering the Event Loop. The only difference is that the FPM, since it does not know the semantics of the event, does not perform specific actions for this event. However, the processing of the event is identical, in that all involved components participate in the event loop in the same way as with 'standard events' (see Reacting to Framework Events).

The following code provides an example of triggering an application-specific event (including event parameters):



### Syntax

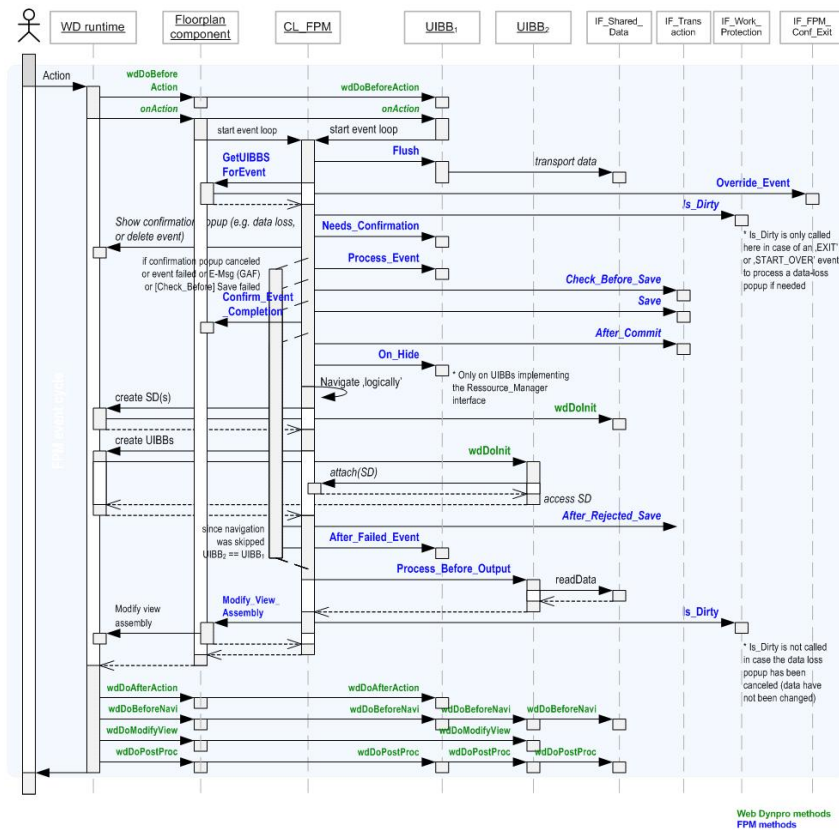
```

1. data: lo_fpm type ref to if_fpm,
2.     lo_event type ref to cl_fpm_event.
3. create object lo_event
4.     exporting
5.     iv_event_id = 'DELETE_AIRPORT'.
6. lo_event->mo_event_data->set_value(
7.     iv_key   = 'AIRPORT_ID'
8.     iv_value = lv_airport_id).
9. lo_fpm = cl_fpm_factory=>get_instance( ).
10. lo_fpm->raise_event( io_event = lo_event ).

```

## Reacting to Framework Events

The FPM event loop is integrated into the Web Dynpro phase model, as the following figure shows:





## Key Web Dynpro Methods

The following Web Dynpro methods are important in FPM applications:

| Method                          | Method Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DOINIT                          | A standard Web Dynpro method that is called only once in the lifetime of a Web Dynpro component by the Web Dynpro runtime. This method is used to initialize your component, e.g. initialize attributes and create helper classes).                                                                                                                                                                                                                                                                                                                           |
| DOBEFOREACTION                  | A standard Web Dynpro method that is called by the Web Dynpro runtime on all visible UIBBs when the user triggers a round trip. According to Web Dynpro programming guidelines, generic validations must be handled in this method; e.g. check that all mandatory fields are filled.                                                                                                                                                                                                                                                                          |
| Action handler<br>(ONACTION...) | The registered Web Dynpro action handler is called. You then have two options: <ul style="list-style-type: none"><li>• If the user interaction does not affect other UIBBs, and there is no need for FPM features such as data-loss dialog boxes, you can handle the action locally in your UIBB. Use standard Web Dynpro programming; e.g. selection of another radio-button leads to different enabled/disabled settings of other controls on the same view.</li><li>• However, for all actions which may affect other UIBBs, raise an FPM event.</li></ul> |



## Different Categories of Web Dynpro Interfaces

Regarding the behavior of instantiating the Web Dynpro components and their participation within the FPM event loop, the Web Dynpro interfaces provided by the FPM can be divided into two categories:

- *Category 1*

More than one Web Dynpro component implements this Web Dynpro interface. Those which do may have more than one instance and the instances may only participate in a part of the FPM event loops during the application's lifetime.

The following Web Dynpro interfaces belong to this category:

- IF\_FPM\_UI\_BUILDING\_BLOCK
- IF\_FPM\_TRANSACTION, IF\_FPM\_WORK\_PROTECTION
- IF\_FPM\_RESOURCE\_MANAGER

- *Category 2*

Only one Web Dynpro component implements this Web Dynpro interface. The corresponding Web Dynpro component has only one instance which participates at all FPM event loops that happen during the application's lifetime.

The following Web Dynpro interfaces belong to this category:

- IF\_FPM\_APP\_CONTROLLER
- IF\_FPM\_SHARED\_DATA
- IF\_FPM\_OIF\_CONF\_EXIT (or IF\_FPM\_GAF\_CONF\_EXIT)



## Generic User Interface Building Block (GUIBB)

You can use Floorplan Manager to compile application-specific views (UIBBs) from one or more applications that were realized as Web Dynpro components into new [Floorplan Manager applications](#). These views generally include the majority of actual applications. Since the views were previously created using the Web Dynpro ABAP foundation, there generally was a high level of variance in the display and navigational behavior of the views. These views cannot be configured in Floorplan Manager.

By introducing generic user interface building blocks, Floorplan Manager has made it possible to improve the uniformity of application-specific views. Generic user interface building blocks are design templates for which, at design time, the application defines the data to be displayed along with a configuration. The concrete display of the data on the user interface is not determined and generated by the GUIBB until runtime. This is done automatically using the configuration provided.

Floorplan Manager provides the following generic user interface building blocks:

- [Form component](#) (Web Dynpro component: FPM\_FORM\_UIBB)
- [List component](#) (Web Dynpro component: FPM\_LIST\_UIBB)
- [Tabbed component](#) (Web Dynpro component: FPM\_TABBED\_UIBB)



## Feeder Classes

A class that implements the IF\_FPM\_GUIBB\_FORM interface (for form components) or the IF\_FPM\_GUIBB\_LIST interface (for list components) and provides all necessary application-specific information to the GUIBB.

### Structure

Using the `GET_DEFINITION` method, the class defines the field catalog of the component and supplies the component at runtime with data from the application using the `GET_DATA` method.

## Features

Feeder class implementations are based on a predefined interface definition providing all necessary methods and corresponding signatures in order to standardize the communication between the application and the GUIBB.

This communication embraces the following:

- Application definition (e.g. data definition, structure or table definitions and their technical aspects)
- Default layout information and corresponding field dependencies
- The (optional) action definition based on metadata
- The action/event handling and data forwarding to the underlying application model



## Form Component (GUIBB FORM)

A generic design template for displaying data in a form that is implemented using the Web Dynpro component `FPM_FORM_UIBB`.

You use this design template in application-specific views (UIBB) where you want to display data using a form. You can determine the concrete display of the data in a form when configuring the Web Dynpro component `FPM_FORM_UIBB`.

## Structure

A FORM is comprised of various subobjects:

- ELEMENT

Elements are descriptor/field combinations that can be configured for the display type of the field or descriptors.

- MELTINGGROUP

A melting group is a group of multiple fields.

- TOOLBAR

Contains buttons that can have actions assigned to them and can be executed in the form.

- GROUP

A group consists of elements, melting groups, and toolbars. You can enter a separate name and group type for each group. The following group types are possible:

- Fullscreen width with one column
- Fullscreen width with two columns
- Half screen width with one column



Note

Only one element or melting group can be displayed per line in a column.



Note

The information that can be displayed on a form is determined by the [feeder class](#) assigned to the configuration of the Web Dynpro component `FPM_FORM_UIBB`.

## Integration

You can configure a form component using the [Form Editor](#) for Floorplan Manager.



### IF\_FPM\_GUIBB\_FORM Interface

The following tables describe the methods (and their attributes) of the `IF_FPM_GUIBB_FORM` interface.

If your application does not need a particular method, implement an empty method, otherwise the system will dump.



Note


You must implement the following methods:

- GET\_DEFINITION
- GET\_DATA


#### Methods

|                                                                                                                                                                                               |  |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| <b>GET_DEFINITION: Allows the feeder to provide all necessary information for configuring a form: the list of available fields and their properties and the list of actions (FPM events).</b> |  |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

| <b>GET_DEFINITION: Allows the feeder to provide all necessary information for configuring a form: the list of available fields and their properties and the list of actions (FPM events).</b>                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameter                                                                                                                                                                                                                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| EO_FIELD_CATALOG                                                                                                                                                                                                                       | <p>Is of type CL_ABAP_STRUCTDESCR. The components of this object are the available fields. The simplest way to provide a field catalog is to create a flat DDIC structure containing all fields and then get the field catalog via</p> <pre>eo_field_catalog ?= CL_ABAP_STRUCTDESCR=&gt;describe_by_name( &lt;name&gt; )</pre> <p> Note</p> <p>The form GUIBB supports only flat structures. When using deep structures, only the highest level fields are available.</p> |
| ET_FIELD_DESCRIPTION                                                                                                                                                                                                                   | Here you can provide the additional information needed to create the form, e.g. Label_by_DDIC, LABEL_REF                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ET_ACTION_DEFINITION                                                                                                                                                                                                                   | A list of all actions (which will be transformed to FPM Events at runtime) that you can assign to form elements.                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| ET_SPECIAL_GROUPS                                                                                                                                                                                                                      | Here you have the same options that you have in the ABAP ALV (see function module REUSE_ALV_GRID_DISPLAY) to group the fields within your field catalogue. You must enter the special group for each field in the field description table in field SP_GROUP. At design-time the FPM Configuration Editor groups the fields. This is an easier way to find fields if your field catalogue contains many fields.                                                                                                                                             |
| <b>GET_PARAMETER_LIST: Called at design time and allows you to define a list of the parameters that the feeder class supports. This list is used by the FPM Configuration Editor to provide the input fields for these parameters.</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Parameter                                                                                                                                                                                                                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| RT_PARAMETER_DESCR                                                                                                                                                                                                                     | Is returned from this method. It describes which parameter is possible. In Field <i>TYPE</i> , the DDIC type needs to be entered.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>INITIALIZE: Called at runtime when the form is created. It is the first feeder method which is called from FPM.</b>                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Parameter                                                                                                                                                                                                                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

| <b>INITIALIZE:</b> Called at runtime when the form is created. It is the first feeder method which is called from FPM.                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                       |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameter                                                                                                                                                                                                                                                                        | Description                                                                                                                                                                                                                                                                                                                                           |
| IT_PARAMETER                                                                                                                                                                                                                                                                     | Contains a list of the feeder parameters and the values for them specified in the configuration.                                                                                                                                                                                                                                                      |
|                                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                       |
| <b>FLUSH:</b> The first feeder method which is called during an event loop. Whenever an FPM event is triggered (this includes all round trips caused by the form itself) this method is called. Use it to forward changed form data to other components in the same application. |                                                                                                                                                                                                                                                                                                                                                       |
| Parameter                                                                                                                                                                                                                                                                        | Description                                                                                                                                                                                                                                                                                                                                           |
| IT_CHANGE_LOG                                                                                                                                                                                                                                                                    | Lists all changes made by the user.                                                                                                                                                                                                                                                                                                                   |
| IS_DATA                                                                                                                                                                                                                                                                          | Is a structure containing the changed data                                                                                                                                                                                                                                                                                                            |
| <b>PROCESS_EVENT:</b> Called within the FPM event loop, it forwards the FPM <b>PROCESS_EVENT</b> to the feeder class. Here the event processing can take place and this is where the event can be canceled or deferred.                                                          |                                                                                                                                                                                                                                                                                                                                                       |
| Parameter                                                                                                                                                                                                                                                                        | Description                                                                                                                                                                                                                                                                                                                                           |
| IO_EVENT                                                                                                                                                                                                                                                                         | The FPM event which is to be processed                                                                                                                                                                                                                                                                                                                |
| EV_RESULT                                                                                                                                                                                                                                                                        | The result of the event processing. There are 3 possible values: <ul style="list-style-type: none"> <li>• <code>ev_result = if_fpm_constants=&gt;gc_event_result-OK</code></li> <li>• <code>ev_result = if_fpm_constants=&gt;gc_event_result-FAILED.</code></li> <li>• <code>ev_result = if_fpm_constants=&gt;gc_event_result-DEFER</code></li> </ul> |
| ET_MESSAGES                                                                                                                                                                                                                                                                      | A list of messages which shall be displayed in the message region.                                                                                                                                                                                                                                                                                    |
| <b>GET_DATA:</b> Called within the FPM event loop and forwards the FPM <b>PROCESS_BEFORE_OUTPUT</b> event to the feeder class. Here you specify the form data after the event has been processed.                                                                                |                                                                                                                                                                                                                                                                                                                                                       |
| Parameter                                                                                                                                                                                                                                                                        | Description                                                                                                                                                                                                                                                                                                                                           |
| IO_EVENT                                                                                                                                                                                                                                                                         | The FPM event which is to be processed.                                                                                                                                                                                                                                                                                                               |
| IT_SELECTED_FIELDS                                                                                                                                                                                                                                                               | The list of fields necessary for the form rendering.                                                                                                                                                                                                                                                                                                  |

| <b>GET_DATA: Called within the FPM event loop and forwards the FPM PROCESS_BEFORE_OUTPUT event to the feeder class. Here you specify the form data after the event has been processed.</b> |                                                                                                                                                                                                                                                                                                                                                             |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Parameter</b>                                                                                                                                                                           | <b>Description</b>                                                                                                                                                                                                                                                                                                                                          |
|                                                                                                                                                                                            | Provide only the data for the fields listed in this table; all other fields are neither visible at runtime nor used as reference for visible fields.                                                                                                                                                                                                        |
| ET_MESSAGES                                                                                                                                                                                | A list of messages which shall be displayed in the message area.                                                                                                                                                                                                                                                                                            |
| EV_DATA_CHANGED                                                                                                                                                                            | For performance reasons, the GUIBB adjusts the data in the form only if the data has been changed. To indicate this, set this flag whenever you change the data to be displayed within this feeder.                                                                                                                                                         |
| EV_FIELD_USAGE_CHANGED                                                                                                                                                                     | Indicates whether or not the field usage has been changed by this method. If you change the field usage without setting this flag to 'X', your changes are ignored.                                                                                                                                                                                         |
| EV_ACTION_USAGE_CHANGED                                                                                                                                                                    | Indicates whether or not the action usage has been changed. Use an 'X' to indicate whether you changed the action usage. If you do not, your changes are ignored.                                                                                                                                                                                           |
| CS_DATA                                                                                                                                                                                    | The form data to be changed.                                                                                                                                                                                                                                                                                                                                |
| CT_FIELD_USAGE                                                                                                                                                                             | Field usage to change. The field usage consists of the field attributes which might change at runtime (e.g. enabled, visible, etc.)<br><br> <b>Note</b><br><br>If you change the fixed values of a field, set the flag <code>FIXED_VALUES_CHANGED</code> for this field. |
| CT_ACTION_USAGE                                                                                                                                                                            | Action usage to change. The action usage consists of the attributes related to actions which might change at runtime. For example, visibility. If an action is rendered as a button, then the visibility setting of the button is defined here.                                                                                                             |

| <b>GET_DEFAULT_CONFIG: Call this if you want to have a default configuration. Use it to call pre-configured form configurations when a user starts the FPM Configuration Editor. This avoids the user, who uses a feeder class to create a form, having to create it from the beginning.</b> |                                                                                                       |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Parameter                                                                                                                                                                                                                                                                                    | Description                                                                                           |
| IO_LAYOUT_CONFIG                                                                                                                                                                                                                                                                             | Of type IF_FPM_GUIBB_FORM_CONFIG: This object provides the API to create a default configuration.     |
| <b>CHECK_CONFIG: Call this if you want to make your own application-specific checks on the configuration in the FPM Configuration Editor immediately before saving.</b>                                                                                                                      |                                                                                                       |
| Parameter                                                                                                                                                                                                                                                                                    | Description                                                                                           |
| IO_LAYOUT_CONFIG                                                                                                                                                                                                                                                                             | Of type IF_FPM_GUIBB_FORM_CONFIG: This object provides the API to read the configuration to be saved. |
| ET_MESSAGES                                                                                                                                                                                                                                                                                  | A list of messages which shall be displayed in the message region.                                    |



## Form Editor for Floorplan Manager

You use the form editor to adjust a form in an application to your specific business requirements. This is done by configuring [form components](#).

### Features

The form editor consists of the following work areas:

- Preview
 

In the preview, all form elements from the current configuration are displayed so as to give you a picture of the layout of the form.
- Hierarchy
 

All form elements (groups, melting groups, and elements) are displayed in the hierarchy.
- Attribute view
 

Attributes of the currently selected form element that can be changed using the form editor are displayed in the attribute view.
- Action area



The action area contains links to all the actions you can execute for the form component. The actions that can be selected depend on the concrete configuration of the form. This means that it can differ within a configuration.

The form editor provides you with the following actions:

- Add Group
- Add Melting Group
- Edit Feeder Class
- Edit Parameters
- Configure Toolbar
- Configure Group

The form editor provides you with the following functions for editing a group:

- Change Group Attributes  
The group name, group type, and index can be changed.
- Add New Group
- Add Melting Group
- Add Element  
You can select a field from the field catalog and determine the label text and display type.
- Delete Group

The form editor provides you with the following functions for editing a melting group:

- Add Group Element  
You can select a field from the field catalog. Fields are configured in more detail by changing the group element attributes.
- Change Group Element Attributes  
The display type, visibility of the label, label text, and index can be changed. Any other group element attributes that can be changed depend on the display type.
- Delete Group Element

The form editor provides you with the following functions for editing a toolbar:

- Add Button
- Change Button

- Delete Button

The form editor is launched in a separate browser window. You can launch the form editor in change or display mode and save your changes at any time.



Note

The component-defined processing view is pre-set. Make sure that this view is selected before configuring a form component.

The form editor launches from the [configuration editor for Floorplan Manager](#) automatically when you launch the configuration of an application-specific view (UIBB) that uses the `FPM_FORM_UIBB` Web Dynpro component.



## Add Form

At any time, you can add a new form as an additional, application-specific view to a Floorplan Manager application. Depending on the floorplan and the application-specific views already embedded, you can position a new form as a [form component](#) in one of the following ways:

- in a subview of an object instance
- in a main step of a guided activity
- in a substep of a guided activity
- in a tabbed component in a master UIBB
- in a tabbed component in a tab UIBB

### Prerequisites

If you would like to add a new form, you must assign a [feeder class](#) to the form. This feeder class must exist in the system.

### Procedure

Substep A: Calling up the [configuration editor](#) of Floorplan Manager

1. Select a Web Dynpro application configuration in the *Object Navigator* of the *ABAP Workbench*.
2. On the *Web Dynpro Explorer: Display Web Dynpro Configuration* screen, choose **► Web Dynpro Configuration → Test → Execute ◀**.

The Web Dynpro application is launched in a separate browser window.

3. In this window, go to the application's identification region and choose the *Adapt Configuration* link.

4. In the *Editor for Web Dynpro ABAP Components — Customizing* screen, choose *Change*.
5. On the *Component Customizing <application name>* screen, make sure that the *Component-Defined* view is on.

#### Substep B: Adding a Form

1. In the preview, select the place where you would like to add the new form.
2. If no UIBB has been defined for a subview or step, choose *Attributes*.
3. If a UIBB has already been assigned, choose *Add UIBB* in the action area.

Now specify the UIBB as a form component by entering the following values in the attribute view:

1. In the *Component* field, enter `FPM_FORM_UIBB`.
2. In the *View* field, enter `FORM_WINDOW`.
3. In the *Configuration Name* field, enter a name for the form component configuration.



#### Note

Although the *Configuration Name* field is not marked as a required entry, it is necessary to enter a name here. Otherwise, the *Configure UIBB* action is not activated.

The *Configuration Type*, *Configuration Variant*, and *Sequence Order* are optional.

4. Choose *Save*.

The system has entered the form component as a new UIBB with the name `UIBB: FORM_WINDOW`.

#### Substep C: Configuring the Form

1. Choose *Configure UIBB*.
2. In the *Editor for Web Dynpro ABAP Components — Configuration* screen, choose *Create*.
3. In the *Create Configuration* dialog window, enter a description and choose a package.
4. Choose *OK*.
5. In the *Edit Feeder Class* dialog window, enter the feeder class that you would like to assign the new form to.



#### Note

You can search for a feeder class here.

To do this, choose the input help in the *Edit Feeder Class* window. The system opens the *Select Feeder Class IF\_FPM\_GUIBB\_FORM* dialog window. You can search for a feeder class here. Choose *Start Search*. The system lists all feeder classes for the `IF_FPM_GUIBB_FORM` Webdynpro interface. Select a feeder class and choose *OK*. The system copies the feeder class into the *Feeder Class* field.

6. Choose *Edit Parameters*.
7. In the *Edit Parameters* window, you can choose a value for the feeder class parameters.
8. Choose *OK*.

## Result

The system opens the [form editor](#) for the new form.



## List Component (GUIBB LIST)

A generic design template for displaying data in a list that is implemented using the Web Dynpro component `FPM_LIST_UIBB`.

You use this design template in application-specific views (UIBB) where you want to display data using a list. You can determine the concrete display of the data in a list when configuring the Web Dynpro component `FPM_LIST_UIBB`.

## Structure

A list consists of a number of columns. The component-defined view gives you the opportunity to specify:

- Which data is displayed in which columns.
- Which display type (such as display field or input field) is used in which column.
- Which order the columns are arranged in.
- The number of columns and rows that can be displayed in the view at one time.



## Note

The data of a list that can be displayed is determined by the [feeder class](#) that is assigned to the configuration of the Web Dynpro component `FPM_LIST_UIBB`.

## Integration

You can configure a list component using the [List Editor](#) for Floorplan Manager.



### IF\_FPM\_GUIBB\_LIST Interface

The following tables describe the methods (and their attributes) of the IF\_FPM\_GUIBB\_LIST interface.

If your application does not need a particular method, implement an empty method, otherwise the system will dump.




#### Note

You must implement the following methods:

- GET\_DEFINITION
- GET\_DATA

#### Methods

| <b>GET_DEFINITION: Allows the feeder to provide all necessary information for configuring a list: the list of available fields and their properties and the list of actions (FPM events).</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Parameter</b>                                                                                                                                                                              | <b>Description</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| EO_FIELD_CATALOG                                                                                                                                                                              | <p>Is of type CL_ABAP_STRUCTDESCR. The components of this object are the available fields. The simplest way to provide a field catalog is to create a flat DDIC structure containing all fields and then get the field catalog via</p> <pre>eo_field_catalog ?=<br/>CL_ABAP_STRUCTDESCR=&gt;describe_by_name( &lt;name&gt; )</pre> <p> Note</p> <p>The list GUIBB supports only flat structures. When using deep structures, only the highest level fields are available.</p> |
| ET_FIELD_DESCRIPTION                                                                                                                                                                          | Here you can provide the additional information needed to create the list, e.g. Label_by_DDIC, LABEL_REF.                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| ET_ACTION_DEFINITION                                                                                                                                                                          | A list of all actions (which will be transformed to FPM events at runtime) that you can assign to list elements.                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

**GET\_DEFINITION:** Allows the feeder to provide all necessary information for configuring a list: the list of available fields and their properties and the list of actions (FPM events).

| Parameter         | Description                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ET_SPECIAL_GROUPS | Here you have the same options that you have in the ABAP ALV (see function module REUSE_ALV_GRID_DISPLAY) to group the fields within your field catalogue. You must enter the special group for each field in the field description table in field SP_GROUP. At design-time the FPM Configuration Editor groups the fields. This is an easier way to find fields if your field catalogue contains many fields. |

**GET\_PARAMETER\_LIST:** Called at design time and allows you to define a list of the parameters that the feeder class supports. This list is used by the FPM Configuration Editor to provide the input fields for these parameters.

| Parameter          | Description                                                                                                               |
|--------------------|---------------------------------------------------------------------------------------------------------------------------|
| RT_PARAMETER_DESCR | Is returned from this method. It describes which parameter is possible. In Field TYPE, the DDIC type needs to be entered. |

**INITIALIZE:** Called at runtime when the list is created. It is the first feeder method which is called from FPM.

| Parameter    | Description                                                                                      |
|--------------|--------------------------------------------------------------------------------------------------|
| IT_PARAMETER | Contains a list of the feeder parameters and the values for them specified in the configuration. |

**FLUSH:** The first feeder method which is called during an event loop. Whenever an FPM event is triggered this method is called (this includes all round trips caused by the list itself). Use it to forward changed list data to other components in the same application.

| Parameter     | Description                                 |
|---------------|---------------------------------------------|
| IT_CHANGE_LOG | Lists all changes made by the user.         |
| IS_DATA       | Is a structure containing the changed data. |

**PROCESS\_EVENT:** Called within the FPM event loop and forwards the FPM PROCESS\_EVENT to the feeder class. Here the event processing can take place and this is where the event can be canceled or deferred.


| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

**PROCESS\_EVENT: Called within the FPM event loop and forwards the FPM PROCESS\_EVENT to the feeder class. Here the event processing can take place and this is where the event can be canceled or deferred.**

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                           |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IO_EVENT    | The FPM event which is to be processed.                                                                                                                                                                                                                                                                                                               |
| EV_RESULT   | The result of the event processing. There are 3 possible values: <ul style="list-style-type: none"> <li>• <code>ev_result = if_fpm_constants=&gt;gc_event_result-OK</code></li> <li>• <code>ev_result = if_fpm_constants=&gt;gc_event_result-FAILED.</code></li> <li>• <code>ev_result = if_fpm_constants=&gt;gc_event_result-DEFER</code></li> </ul> |
| ET_MESSAGES | A list of messages which shall be displayed in the message region.                                                                                                                                                                                                                                                                                    |

**GET\_DATA: Called within the FPM event loop, it forwards the FPM PROCESS\_BEFORE\_OUTPUT event to the feeder class. Here you specify the list data after the event has been processed.**

| Parameter               | Description                                                                                                                                                                                               |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IO_EVENT                | The FPM event which is to be processed                                                                                                                                                                    |
| IT_SELECTED_FIELDS      | The list of fields necessary for the list rendering. Provide only the data for the fields listed in this table; all other fields are neither visible at runtime nor used as reference for visible fields. |
| ET_MESSAGES             | A list of messages which shall be displayed in the message area.                                                                                                                                          |
| EV_DATA_CHANGED         | For performance reasons, the GUIBB adjusts the data in the list only if the data has been changed. To indicate this, set this flag whenever you change the data to be displayed within this feeder.       |
| EV_FIELD_USAGE_CHANGED  | Indicates whether or not the field usage has been changed by this method. If you change the field usage without setting this flag to 'X', your changes are ignored.                                       |
| EV_ACTION_USAGE_CHANGED | Indicates whether or not the action usage has been changed. Use an 'X' to indicate whether you changed the action usage. If you do not, your changes are ignored.                                         |

| <b>GET_DATA: Called within the FPM event loop, it forwards the FPM PROCESS_BEFORE_OUTPUT event to the feeder class. Here you specify the list data after the event has been processed.</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameter                                                                                                                                                                                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| CS_DATA                                                                                                                                                                                    | The list data to be changed.                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| CT_FIELD_USAGE                                                                                                                                                                             | Field usage to change. The field usage consists of the field attributes which might change at runtime (e.g. enabled or disabled, visible or invisible, mandatory or optional, read-only or edit). Use it to control the properties of columns.<br><br> Note<br><br>If you change the fixed values of a field, set the flag <code>FIXED_VALUES_CHANGED</code> for this field. |
| CT_ACTION_USAGE                                                                                                                                                                            | Action usage to change. The action usage consists of the attributes related to actions which might change at runtime (e.g. enabled or disabled, visible or invisible, mandatory or optional, read-only or edit). Use it to control the properties of toolbars. If an action is rendered as a button, the visibility setting (for example) of the button is defined here.                                                                                      |

 Note

Regarding columns, note that it is possible to assign the attributes for `CT_FIELD_USAGE` and `CT_ACTION_USAGE` either to single cells or to whole columns (with the exception of visibility, which can be applied only to whole columns).

If you want to set these attributes for the *whole column*, use the corresponding fields in the `field_usage` structure.

If you would like to set these attributes for *single cells*, proceed as follows:

1. Create a new column for the table (add a field to the field catalog in the `GET_DEFINITION` method).
2. Define the column as a technical column that is not visible at runtime, by setting the field `technical_field` to 'X'. This column contains the properties of the cells.
3. In the `GET_DEFINITION` method, adjust the field description accordingly. For example, you have a column A and you want to set the property `Read_only` for single cells in that column. For this reason you created a technical column B. In the field description, set `read_only_ref` to B.



| <b>GET_DEFAULT_CONFIG: Call this if you want to have a default configuration. Use it to call pre-configured list configurations when a user starts the FPM Configuration Editor. This avoids the user, who uses a feeder class to create a list, having to create it again from the beginning.</b> |                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| Parameter                                                                                                                                                                                                                                                                                          | Description                                                                                            |
| IO_LAYOUT_CONFIG                                                                                                                                                                                                                                                                                   | Of type IF_FPM_GUIBB_LIST_CONFIG: This object provides the API to create a default configuration.      |
| <b>CHECK_CONFIG: Call this if you want to make your own application-specific checks on the configuration in the FPM Configuration Editor immediately before saving.</b>                                                                                                                            |                                                                                                        |
| Parameter                                                                                                                                                                                                                                                                                          | Description                                                                                            |
| IO_LAYOUT_CONFIG                                                                                                                                                                                                                                                                                   | Oof type IF_FPM_GUIBB_LIST_CONFIG: This object provides the API to read the configuration to be saved. |
| ET_MESSAGES                                                                                                                                                                                                                                                                                        | A list of messages which shall be displayed in the message region.                                     |



## List Editor for Floorplan Manager

You use the list editor to adjust a list within an application to you specific business requirements. This is done by configuring [list components](#).

### Features

The list editor consists of the following work areas:

- Preview

In the preview, the list in the current configuration is displayed so as to give you a picture of the layout of the list.

- Hierarchy

All list elements (columns, toolbar, and parameters) are displayed in the hierarchy as a tree structure.

- Attribute view

Attributes of the currently selected list element that can be changed using the list editor are displayed in the attribute view.

- Action area

The action area contains links to all the actions you can execute for the list component. Which actions can be selected depends on the concrete configuration of the list. This means that the selection of actions can differ within a configuration.

The list editor provides you with the following actions:

- Edit Feeder Class
- Edit Parameters
- Configure Column
- Configure Toolbar

The form editor provides you with the following functions for editing a column:

- Add Column

You can select a field from the field catalog and determine the column header and display type.

- Delete Column

The list editor is launched in a separate browser window. You can launch the list editor in change or display mode and save your changes at any time.



Note

The component-defined processing view is pre-set. Make sure that this view is selected before configuring a list component.

The list editor launches from the [configuration editor for Floorplan Manager](#) automatically when you launch the configuration of an application-specific view (UIBB) that uses the `FPM_LIST_UIBB` Web Dynpro component.



## Add List

You can add an additional, application-specific view to a Floorplan Manager application at any time in the form of a new list. Depending on the floorplan and the application-specific views already embedded, you can position a new [list component](#) in one of the following ways:

- in a subview of an object instance
- in a main step of a guided activity
- in a substep of a guided activity
- in a tabbed component in a master UIBB

- in a tabbed component in a tab UIBB

## Prerequisites

If you would like to add a new form, you must assign a [feeder class](#) to the form. This feeder class must first be created and programmed.

## Procedure

Substep A: Calling up the [configuration editor](#) of Floorplan Manager

1. Select a Web Dynpro application configuration in the *Object Navigator* of the *ABAP Workbench*.
2. On the *Web Dynpro Explorer: Display Web Dynpro Configuration* screen, choose ► *Web Dynpro Configuration* → *Test* → *Execute* ◀.

The Web Dynpro application is launched in a separate browser window.

3. In this window, go to the application's identification region and choose the *Adapt Configuration* link.
4. In the *Editor for Web Dynpro ABAP Components — Customizing* screen, choose *Change*.
5. On the *Component Customizing <application name>* screen, make sure that the *Component-Defined* view is on.

Substep B: Adding a List

1. In the preview, select the place where you would like to add a new list.
2. If no UIBB has been defined for a subview or step, choose *Attributes*.
3. If a UIBB has already been assigned, choose *Add UIBB* in the action area.

Now specify the UIBB as a list component by entering the following values in the attribute view:

1. In the *Component* field, enter `FPM_LIST_UIBB`.
2. In the *View* field, enter `LIST_WINDOW`.
3. In the *Configuration Name* field, enter a name for the list component configuration.



### Note

Although the *Configuration Name* field is not marked as a required entry, it is necessary to enter a name here. Otherwise, the *Configure UIBB* action is not activated.

The *Configuration Type*, *Configuration Variant*, and *Sequence Order* are optional.

4. Choose *Save*.

The system has entered the list component as a new UIBB with the name  
UIBB: LIST\_WINDOW.

#### Substep C: Configuring the List

1. Choose *Configure UIBB*.
2. In the *Editor for Web Dynpro ABAP Components — Configuration* screen, choose *Create*.
3. In the *Create Configuration* dialog window, enter a description and choose a package.
4. Choose *OK*.
5. In the *Edit Feeder Class* dialog window, enter the feeder class that you would like to assign the new list to.



#### Note

You can search for a feeder class here.

To do this, choose the input help in the *Edit Feeder Class* window. The system opens the *Select Feeder Class IF\_FPM\_GUIBB\_LIST* dialog window. You can search for a feeder class here. Choose *Start Search*. The system lists all feeder classes for the IF\_FPM\_GUIBB\_LIST Webdynpro interface. Select a feeder class and choose *OK*. The system copies the feeder class into the *ObjectName* field.

6. Choose *Edit Parameters*.
7. In the *Edit Parameters* window, you can choose a value for the feeder class parameters.
8. Choose *OK*.

#### Result

The system opens the [list editor](#) for the new list.



#### Additional Information on the List Component

The following information is useful when configuring a List Component.

#### Attributes

In the hierarchy of the Component Configuration of your application, the following attribute is available for the List Component:

- *Lead Selection Action Assignment:* You can assign an FPM event ID to the lead selection here. If a lead selection occurs during runtime, the assigned FPM event is raised. If you assign no event ID, the generic event ID `IF_FPM_GUIBB_LIST=>GC_FPM_EVENT_ON_LEAD_SEL` is assigned.

In the hierarchy of the Component Configuration of your List Component, choose *Settings* to display the following attributes:

- *Column count:* Determines the amount of columns that are displayed at runtime
- *Row count:* Determines the amount of rows that are displayed at runtime
- *Selection Event:* Like a Web Dynpro table, the List Component offers two kinds of selection at runtime:
  - *Lead selection* (the user uses the left mouse button to select one single row)
  - *Normal selection* (the user uses the right mouse button to select one or more rows)

Using this dropdown list box, you can determine what kind of selection raises an FPM event. The default is a Lead Selection.

- *Selection Mode:* Determines whether it is possible to select multiple rows
- *Selection behavior:* Determines whether currently selected rows are de-selected when the user makes a new selection

#### **FPM Events and the List Component**

As the List Component is itself an FPM UIBB, it takes part, when it is visible, in each FPM event loop. The List Component may also raise FPM events itself. These events are raised from the following three sources:

- Cell events

The columns may contain fields that have a display type that are capable of raising an event (for example, a button display type). All cell-based events have the FPM event ID

`IF_FPM_GUIBB_LIST=>GC_GUIBB_LIST_ON_CELL_ACTION`. The corresponding row and column values are added as event parameters to this FPM event `IF_FPM_GUIBB_LIST=>GC_EVENT_PAR_ROW` and `IF_FPM_GUIBB_LIST=>GC_EVENT_PAR_COLUMN_NAME`.

- Toolbar events

Almost each toolbar element may raise an FPM event. In this case, the event ID is the action ID (which was defined by the feeder class in method `get_definition`). Some toolbar elements may contain specific values of interest (i.e. e. user inputs), such as the toggle button, the input field and the dropdown list box. To get these values, you may read the following FPM event parameters `IF_FPM_GUIBB_LIST=>GC_EVENT_PAR_TOGGLE_STATE` (for the

toggle button), `IF_FPM_GUIBB_LIST=>GC_EVENT_PAR_INPUT_VALUE` (for the input field) or `IF_FPM_GUIBB_LIST=>GC_EVENT_PAR_DROP_DOWN_KEY` (for the dropdown list box).

- Selection events

A row selection may also raise an FPM event. It is possible to choose whether only a lead selection raises an FPM event or also a normal selection (see configuration settings for details).



## Tabbed Component (GUIBB TABBED COMPONENT)

A generic design template for organizing additional application-specific views (UIBB) as tabs that is implemented using the Web Dynpro component `FPM_TABBED_UIBB`.

You use this design template for an application-specific view (UIBB) For example, you could use the template where you want to simultaneously display a selection list of business objects and the additional details of those business objects in tabs without changing the view. You can determine the concrete arrangement of the selection list, detail views, and data when configuring the Web Dynpro component `FPM_TABBED_UIBB`.

### Structure

A tabbed component consists of two areas: the `MASTER` area and the `TAB` area, which can be arranged next to or on top of one another. If you arrange the areas horizontally, the master area is placed to the left of the tab area. If you arrange the areas vertically, the master area is placed above the tab area.

The content of the master area and the content of the tabs are determined by separate Web Dynpro components, which you set when configuring the Web Dynpro component `FPM_TABBED_UIBB`.



### Note

If you do not set the Web Dynpro component for the master area, this area is not displayed in the application. Instead, only the tabs appear with their application-specific views.



## Tabbed Component Editor for Floorplan Manager

You use this editor to adjust a [tabbed component](#) within an application to your specific business requirements. This is done by configuring the component.

## Features

The editor consists of the following work areas:

- Preview

In the preview, all application-specific views (UIBBs) from the current configuration are displayed so as to give you a picture of the layout of the tabbed component.

- Layout

In this area, you determine whether the tabbed component should be arranged horizontally or vertically.

- Hierarchy

All application-specific views (UIBBs) are displayed in the hierarchy as a tree structure.

- Attribute view

Attributes of the currently selected application-specific view (UIBB) that can be changed using the editor are displayed in the attribute view.

- Action area

The action area contains links to all the actions you can execute for the tabbed component.

The editor for a tabbed component provides you with the following actions:

- Add Master Component (technical name: MASTER UIBB)
- Add Tab (technical name: TAB)
- Add Application-Specific View to Tab (technical name: TAB UIBB)

The editor for a tabbed component is launched in a separate browser window. You can launch the editor in change or display mode and save your changes at any time.



Note

The component-defined processing view is pre-set. Make sure that this view is selected before configuring a tabbed component.

The editor for a tabbed component launches from the [configuration editor for Floorplan Manager](#) automatically when you launch the configuration of an application-specific view (UIBB) that has the `FPM_TABBED_UIBB` Web Dynpro component.



### Add Tabbed Component

You can add an additional, application-specific view to a Floorplan Manager application at any time in the form of a new tabbed component. Depending on the floorplan and the application-specific views already embedded, you can position a new [Tabbed Component](#) in one of the following ways:

- in a subview of an object instance
- in a main step of a guided activity
- in a substep of a guided activity
- in a tabbed component in a master UIBB
- in a tabbed component in a tab UIBB

### Procedure

Substep A: Calling up the [configuration editor](#) of Floorplan Manager

1. Select a Web Dynpro application configuration in the *Object Navigator* of the *ABAP Workbench*.
2. In the *Web Dynpro Explorer: Display Web Dynpro Configuration* screen, choose ► *Web Dynpro Configuration* → *Test* → *Execute in Administrator Mode* ↵.

The Web Dynpro application is launched in a separate browser window.

3. In this window, go to the application's identification region and choose the *Adapt Configuration* link.
4. In the *Editor for Web Dynpro ABAP Components — Customizing* screen, choose *Change*.
5. On the *Component Customizing <application name>* screen, make sure that the *Component-Defined* view is on.

Substep B: Adding a Tabbed Component

1. In the preview, select the place where you would like to add a new tabbed component.
2. If no UIBB has been defined for a subview or step, choose *Attributes*.
3. If a UIBB has already been assigned, choose *Add UIBB* in the action area.

Now specify the UIBB as a tabbed component by entering the following values in the attribute view:

1. In the *Component* field, enter `FPM_TABBED_UIBB`.
2. In the *View* field, enter `TABBED_WINDOW`.



3. In the *Configuration Name* field, enter a name for the tabbed component configuration.



Note

Although the *Configuration Name* field is not marked as a required entry, it is necessary to enter a name here. Otherwise, the *Configure UIBB* action is not activated.

The *Configuration Type*, *Configuration Variant*, and *Sequence Order* are optional.

4. Choose Save.

The system has entered the tabbed component as a new UIBB with the name `UIBB: TABBED_WINDOW`.

#### Substep C: Configuring the Tabbed Component

1. Choose *Configure UIBB*.
2. In the *Editor for Web Dynpro ABAP Components — Configuration* screen, choose *Create*.
3. In the *Create Configuration* dialog window, enter a description and choose a package.
4. Choose *OK*.

#### Result

The system opens the [editor for tabbed components](#). In the preview, a tab is displayed that the system has generated.



#### Changing the Tabbed Component Dynamically at Runtime

You may rename, add and remove tabs or child-UIBBs (or embedded UIBBs) from your tabbed component during runtime.

To do so, proceed as follows:

1. Choose an application-specific Web Dynpro component and add the Web Dynpro interface `IF_FPM_TABBED_CONF_EXIT` to the Implemented Interfaces tab of your Web Dynpro component. This is one of the Web Dynpro components that provide you with a child UIBB.
2. Save and activate the newly added interface.

*Example:* Somewhere in your code you want to rename a tab. To do this, you must raise your own FPM event (e. g. `CHANGE_TAB_NAME`) as the sample code below shows:



### Syntax

```

1.      DATA: lo_fpm TYPE REF TO if_fpm,
2.          lo_event TYPE REF TO cl_fpm_event.
3.      lo_fpm = cl_fpm=>get_instance( ).
4.      lo_event = cl_fpm_event=>create_by_id(
      'CHANGE_TAB_NAME' ).
5.      lo_event->mo_event_data->set_value( iv_key   = 'ID'
6.      lo_event->mo_event_data->set_value( iv_key   =
      'NAME'
7.  iv_value =
      lv_tab_name ).
8.      lo_fpm->raise_event( io_event = lo_event ).

```

3. In the component controller, implement the method `OVERWRITE_CONFIG_TABBED`. To continue with the above example of renaming a tab, implement the following sample code:



### Syntax

```

1.      CASE io_tabbed->mo_event->mv_event_id.
2.          WHEN 'CHANGE_TAB_NAME'.
3.              DATA lv_name TYPE string.
4.              DATA lv_id TYPE string.
5.              io_tabbed->mo_event->mo_event_data->get_value(
      EXPORTING iv_key   = 'ID'
6.
      IMPORTING ev_value = lv_id ).
7.              io_tabbed->mo_event->mo_event_data->get_value(
      EXPORTING iv_key   = 'NAME'
8.
      IMPORTING ev_value = lv_name ).
9.              io_tabbed->rename_tab( iv_tab_id   = lv_id
10.                                  iv_new_name = lv_name ).

```



## Navigation

To navigate to a specific application outside of your FPM application, you use the following FPM toolbar menus:

- *You Can Also*
- *Related Links*

These FPM toolbar menus utilize [launchpads](#).

### Navigation APIs

The FPM also provides you with the following two navigation interfaces, allowing you to control the launchpads:

- [IF FPM NAVIGATION](#): Use this to navigate to an application using a given launchpad.

- [IF FPM NAVIGATE TO](#): Use this to navigate to an application without using a launchpad.

### Suspend and Resume

From Enhancement Package 1 of NW onwards, the Suspend and Resume feature is available for FPM applications. This can be described briefly as a feature in which the Web Dynpro application, built within the FPM framework, can be placed in a suspended state whilst the user navigates to another URL. The user can work on the URL and then navigate back to the suspended FPM application, which is resumed from exactly the same state before navigation occurred.

Note that the usage of a report launchpad is mandatory to enable suspend and resume for FPM applications.

For a detailed explanation of this feature, see [Suspend and Resume](#).



### Launchpad

A collection of navigation destinations that are stored as a separate technical object in the system.

You use a launchpad to allow users to navigate to specific goals outside of the current [Floorplan Manager application](#). For example, this could mean navigating to other Web Dynpro ABAP applications, external Web pages, transactions, reports, or other business objects. Within a floorplan, two elements, `YouCanAlso` and `RelatedLinks`, are available to you in the toolbar. You can assign different launchpads to these elements.



#### Note

The launchpad is displayed as a dropdown list. Descriptions that you have written for launchpad applications cannot be displayed in Floorplan Manager.

`YouCanAlso` and `RelatedLinks` are default toolbar elements of the SAP user interface design for floorplans. They are displayed as links. In the concrete Floorplan Manager application, both elements can have different names on the user interface. This name can also be changed using the configuration editor.

### Structure

Any number of applications can be assigned to a launchpad. Each application is described by its application type as well as additional attributes that are dependent on the application type. In a Floorplan Manager application, not every application type can be called up in every portal. The following table describes which application types can be called up in which portal.

| <i>Application Type</i>      | <i>SAP Enterprise Portal</i> | <i>SAP NetWeaver Business Client (Portal Roles)</i> | <i>SAP NetWeaver Business Client (PFCG Roles)</i> |
|------------------------------|------------------------------|-----------------------------------------------------|---------------------------------------------------|
| Portal Pages                 | Yes                          | Yes                                                 | No                                                |
| Transactions                 | Yes                          | Yes                                                 | Yes                                               |
| Report Writer                | Yes                          | Yes                                                 | Yes                                               |
| URL                          | Yes                          | Yes                                                 | Yes                                               |
| SAP BI Report (Query)        | Yes                          | Yes                                                 | No                                                |
| SAP BI Report (Web Template) | Yes                          | Yes                                                 | No                                                |
| BI Report                    | Yes                          | Yes                                                 | No                                                |
| BEx Analyzer                 | Yes                          | Yes                                                 | Yes                                               |
| Manager's Desktop            | Yes                          | Yes                                                 | Yes                                               |
| Web Dynpro Java              | Yes                          | Yes                                                 | No                                                |
| Web Dynpro ABAP              | Yes                          | Yes                                                 | Yes                                               |
| KM Document                  | Yes                          | Yes                                                 | No                                                |
| Visual Composer xApps        | Yes                          | Yes                                                 | No                                                |
| Info Set Query               | Yes                          | Yes                                                 | Yes                                               |
| Object-Based Navigation      | Yes                          | Yes                                                 | Yes                                               |

Multiple folders can be created for every launchpad. You can use these folders to group applications.



#### Note

As opposed to the conventional use of launchpads in portals, in Floorplan Manager, only those navigation destinations are shown that were created as applications in the first folder of the launchpad. The applications in the other folders of the launchpad are hidden in the display.

## Create a Launchpad with Applications

### Procedure

To create a launchpad, complete the following steps:

1. On the *SAP Easy Access* screen, enter `/nlpd_cust` in the command field.
2. Choose *Continue*.  
The system calls up the start screen for Launchpad Customizing.
3. On the *Overview of Launchpads* screen, choose *New Launchpad*.
4. In the dialog box, enter a role name, an instance name, and a description.
5. Choose *Continue*.
6. On the *Change Launchpad Role: <name of role> Instance: <name of instance>*, choose *New Application*.
7. Enter the data for the linktext.



#### Note

You can write a description for the application. However, Floorplan Manager cannot display this description in the dropdown list.

8. Choose an application type.
9. Enter the information for the selected application type.
10. Choose *Save*.

To add multiple applications, repeat steps 6 through 9.



#### Recommendation

For launchpads to which you would like to assign an application, we recommend that you do not use a folder. This is because Floorplan Manager can currently only display one folder. If the launchpad consists of multiple folders, only the topmost folder is displayed.

## Include a Launchpad in the User Interface

### Procedure

To assign a launchpad to the `YouCanAlso` or `RelatedLink` elements on the user interface, complete the following steps.

1. Select a Web Dynpro application and application component.
2. Select a Web Dynpro application configuration in the *Object Navigator* of the *ABAP Workbench*.
3. On the *Web Dynpro Explorer: Display Web Dynpro Configuration* screen, choose ► *Web Dynpro Configuration* → *Test* → *Execute in Administrator Mode* ◀.

The Web Dynpro application is launched in a separate browser window.

4. In this window, go to the application's identification region and choose the *Adapt Configuration* link.
5. On the *Editor for Web Dynpro ABAP Components — Customizing* screen, choose *Change*.
6. On the *Component Customizing <application name>* screen, make sure that the *Component-Defined* view is on.
7. In the navigation area of the configuration editor, choose the *Toolbar* element.
8. Choose *Expand Node*.
9. To display the attributes of the `YouCanAlso` element, click the `YouCanAlso` pushbutton.

If you would like to assign the launchpad to the `RelatedLinks` element, choose the `RelatedLinks` element.

10. In the *Role* field, enter the name of the launchpad role.
11. In the *Instance* field, enter the name of the launchpad instance.
12. To change the name of the button element, enter a different name in the *Name* field.
13. Save the configuration.
14. Test the new configuration.



## Working in the Navigation Customizing

### General Settings

You can make changes to a launchpad that are valid for all destinations and applications in a launchpad. To do this, proceed as follows:

1. Choose transaction `LPD_CUST`.
2. Locate and open your launchpad.
3. On the menu bar, choose ► *Extras* → *General Settings* ◀.

4. In the *Change Launchpad Role* dialog box you can set the following options:

- *OBN Navigation Mode*
  - *User Set of Roles*: An Object Based Navigation (OBN) target can be assigned to any roles that are assigned to a user.
  - *Source Role*: The navigation target must be assigned to the same role as the application that uses the launchpad.
- *The BI Access Type* (defines the data source used by a BI application)
  - *BI System Default*: The data source is determined by the BI system itself.
  - *Access to replicated data*: The data source is the BI system.
  - *Direct access to operative data*: The data source is an ECC system.
- *Check Application Alias is Unique*

Use this to check that a destination application alias being used by an FPM application is unique

#### **Source Parameters and Parameter Mapping**

To support parameter mapping, you can define a set of parameters that are known by the FPM application that uses the launchpad. To do this, proceed as follows:

To do this, proceed as follows:

1. Choose transaction `LPD_CUST`.
2. Locate and open your launchpad.
3. On the menu bar, choose **► Extras → Source Parameters ◀**. The *Default Parameters* dialog box displays the parameters which are known by your FPM application.
4. Some application categories (e.g. Web Dynpro ABAP) have a button *Parameter Mapping* in the *Parameter* input field. To map your parameters, choose this button to extend the *Default Parameter* dialog box and enter the following data:
  - *Parameter*: Enter the parameter name that the FPM application sends to the destination application.
  - *Replaced by*: Enter the parameter name that the destination application expects to receive. When you launch the destination application, the launchpad automatically replaces the parameter that was sent by the FPM application by the parameter in the *Replaced by* column.

#### **Copying an entire Launchpad**

To copy an entire launchpad, proceed as follows:

1. Choose transaction `LPD_CUST`.
2. On the menu bar of the *Overview of Launchpads* screen, choose **► Launchpad** → *Read from other system by RFC* **◄**.
3. In the *Action Launchpad* dialog box, enter an RFC destination. This displays a list of launchpads available in the system and client you have entered as the RFC destination. Note: If you leave *RFC-Destination* empty, you are provided with a list of all launchpads in your current client.
4. Choose *Continue*.
5. Choose the launchpad that you want to copy and choose *Continue*.
6. In the dialog box that appears, enter another role and/or instance. Choose *Continue*.

#### **Copying Applications from one Launchpad to another Launchpad**

For convenience, you can copy one or more applications inside a launchpad to another launchpad. To do this, proceed as follows:

1. Choose transaction `LPD_CUST`.
2. Choose *New Launchpad*.
3. In the dialog box, enter the *Role*, *Instance* and *Description*. Choose *Continue*.
4. Choose *Copy from other Launchpad*. A dialog box appears, listing all the launchpads in the current system and client.
5. Choose a Launchpad and choose *Continue*.
6. In the navigation area, your chosen launchpad appears next to your new launchpad. Click and drag the launchpad destination applications from your chosen launchpad to your new launchpad. Note that you can also drag an entire folder to a new launchpad.
7. Save your entry.

#### **Performing Searches in Launchpads of a Client**

1. You can perform searches covering all launchpads in a system and client. To do this, proceed as follows:
2. To initialize a search, run the report `APB_LPD_UPDATE_SEARCH_TABLE`. This report analyzes all launchpads in a system and client, and displays the information in a search table.
3. In the Launchpad Customizing, *Change Launchpad Role* screen, choose *Search*.
4. You are provided with a list containing all applications of all launchpads in the current client. You can now perform searches throughout the whole list and select the appropriate applications or launchpads.



5. Choose *OK* to display a selected application in the navigation area.

### Re-Displaying a SAP-Delivered Launchpad

You can make changes to a particular launchpad and save the changes. However, it is still possible to display the SAP-delivered launchpad. To do this, proceed as follows:

1. Choose transaction `LPD_CUST`.
2. Open the launchpad which you have made changes to.
3. Choose *Extras* → *Show SAP Version* . In the navigation area, the SAP-delivered launchpad (without the changes) appears alongside the changed version of the launchpad.

### Transporting a Launchpad

To transport a launchpad, proceed as follows:

1. Choose transaction `LPD_CUST`.
2. Open the launchpad you want to transport in change mode.
3. Choose *Launchpad* → *Transport* .
4. In the dialog box, enter the package to which you want to assign the texts that you created in the launchpad. As a result, the texts are also forwarded to translation. Choose *Continue*.
5. In the dialog box, enter a *Customizing request* and choose *Continue*. This request includes the relevant table entries for the following tables:
  - `APB_LAUNCHPADT`
  - `APB_LAUNCHPAD_V`
  - `APB_LPD_CONTROL`
  - `APB_LPD_OTR_KEYS`
  - `APB_LPD_VERSIONS`
6. In the dialog box, enter a *Workbench request*. This request includes the texts from the launchpad. These are objects of the type `R3TR DOCT`.
7. Release both requests.



### **IF\_FPM\_NAVIGATION API (Runtime class CL\_FPM\_NAVIGATION)**

This navigation interface provides you with a list, `MT_TARGETS`, with all customized applications of a given launchpad.

To access this Navigation API, use the interface `IF_FPM`. This provides the `GET_NAVIGATION` method, which returns an instance of the Navigation API, `IF_FPM_NAVIGATION`.

### Tables and Domains

| <b>Table: MT_TARGETS</b>                  |                               |                            |                                                                                              |
|-------------------------------------------|-------------------------------|----------------------------|----------------------------------------------------------------------------------------------|
| <b>Parameters</b>                         | <b>Type kind</b>              | <b>Type</b>                | <b>Description</b>                                                                           |
| entry_type                                | Type                          | FPM_NAVIGATION_TARGET_TYPE | entry_type Type<br>FPM_NAVIGATION_TARGET_TYPE<br>Type of application.                        |
| parent                                    | Type                          | STRING                     | GUID of the parent folder or initial.                                                        |
| key                                       | Type                          | STRING                     | Key Type String GUID of application.                                                         |
| alias                                     | Type                          | STRING                     | A (unique) identifier for an application. It is defined in the customizing of the launchpad. |
| text                                      | Type                          | TEXT255                    | Text of the link.                                                                            |
| description                               | Type                          | STRING                     | Description.                                                                                 |
| icon_path                                 | Type                          | STRING                     | Path to an icon.                                                                             |
| enable                                    | Type                          | BOOLE_D                    | Determines if an application is active/enabled or inactive/disabled.                         |
| visible                                   | Type                          | BOOLE_D                    | Determines the visibility of an application                                                  |
| <b>Domain: FPM_NAVIGATION_TARGET_TYPE</b> |                               |                            |                                                                                              |
| <b>ID</b>                                 | <b>Description</b>            |                            |                                                                                              |
| APP                                       | Line contains an application. |                            |                                                                                              |
| FOL                                       | Line contains a folder.       |                            |                                                                                              |
| SEP                                       | Line contains a separator.    |                            |                                                                                              |

### Methods

This navigation interface provides the methods described in the tables below:

| <b>NAVIGATE: Starts the navigation of an application.</b>                                                                                                                                |           |           |                  |                                                                            |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|-----------|------------------|----------------------------------------------------------------------------|
| Parameters                                                                                                                                                                               | Direction | Type kind | Type             | Description                                                                |
| IV_TARGET_KEY                                                                                                                                                                            | importing | Type      | STRING           | GUID of Application.                                                       |
| <b>MODIFY: Changes attributes of an application. For example, you can change the visibility of an application, enable or disable an application and change its description and text.</b> |           |           |                  |                                                                            |
| Parameters                                                                                                                                                                               | Direction | Type kind | Type             | Description                                                                |
| IV_VISIBLE                                                                                                                                                                               | importing | Type      | BOOLE_D          | Set an application to visible/invisible.                                   |
| IV_ENABLE                                                                                                                                                                                | importing | Type      | BOOLE_D          | Enable/disable an application.                                             |
| IV_TEXT                                                                                                                                                                                  | importing | Type      | STRING           | An alternative text for the application.                                   |
| IV_DESCRIPTION                                                                                                                                                                           | importing | Type      | STRING           | An alternative description for the application.                            |
| IV_TARGET_KEY                                                                                                                                                                            | importing | Type      | STRING           | GUID of Application.                                                       |
| IV_NOTIFY                                                                                                                                                                                | importing | Type      | BOOLE_D          | Invokes notification on all registered nodes / objects.                    |
| <b>SET_FILTER: Allows you to remove some applications from the list of applications that the launchpad provides.</b>                                                                     |           |           |                  |                                                                            |
| Parameters                                                                                                                                                                               | Direction | Type kind | Type             | Description                                                                |
| IT_Filter                                                                                                                                                                                | importing | Type      | T_FILTER         | GUIDs of application that must not be provided in the list of application. |
| <b>MODIFY_PARAMETERS: Changes the values of existing parameters or adds a parameter if none exists.</b>                                                                                  |           |           |                  |                                                                            |
| Parameters                                                                                                                                                                               | Direction | Type kind | Type             | Description                                                                |
| ID_TARGET_KEY                                                                                                                                                                            | importing | Type      | STRING           | GUID of Application.                                                       |
| IT_APPLICATION_PARAMETER                                                                                                                                                                 | importing | Type      | APB_LPD_T_PARAMS | Contains application                                                       |

| <b>MODIFY_PARAMETERS: Changes the values of existing parameters or adds a parameter if none exists.</b> |                  |                  |                     |                                                                                               |
|---------------------------------------------------------------------------------------------------------|------------------|------------------|---------------------|-----------------------------------------------------------------------------------------------|
| <b>Parameters</b>                                                                                       | <b>Direction</b> | <b>Type kind</b> | <b>Type</b>         | <b>Description</b>                                                                            |
|                                                                                                         |                  |                  |                     | parameters that will be added or changed.                                                     |
| IT_BUSINESS_PARAMETER                                                                                   | importing        | Type             | APB_LPD_T_PARAMS    | Contains business parameters that will be added or changed.                                   |
| <b>ADD_BEX_ANALYZER: Adds an application of type BEx Analyzer to a given launchpad.</b>                 |                  |                  |                     |                                                                                               |
| <b>Parameters</b>                                                                                       | <b>Direction</b> | <b>Type kind</b> | <b>Type</b>         | <b>Description</b>                                                                            |
| IV_PARENT_FOLDER_ID                                                                                     | importing        | Type             | FPM_APPLICATION_ID  | GUID of parent folder. If the parameter is empty, the application will be added at top level. |
| IS_BEX_ANALYZER_FIELDS                                                                                  | importing        | Type             | FPM_S_BEX_ANALYZER  | Structure that contains the fields to add with BEx Analyzer application type.                 |
| EV_APPLICATION_ID                                                                                       | exporting        | Type             | FPM_APPLICATION_ID  | GUID of Application.                                                                          |
| ET_MESSAGES                                                                                             | exporting        | Type             | FPM_T_T100_MESSAGES | Error messages.                                                                               |

| <b>ADD_BEX_ANALYZER: Adds an application of type BEx Analyzer to a given launchpad.</b> |                  |                  |             |                                                                                |
|-----------------------------------------------------------------------------------------|------------------|------------------|-------------|--------------------------------------------------------------------------------|
| <b>Parameters</b>                                                                       | <b>Direction</b> | <b>Type kind</b> | <b>Type</b> | <b>Description</b>                                                             |
| EV_ERROR                                                                                | exporting        | Type             | BOOLE_D     | Status = false - the application was added; Status = true - an error occurred. |

The following are other methods with a similar interface to ADD\_BEX\_ANALYZER, which allow you to add a specified application, at runtime, to a launchpad:

- ADD\_URL
- ADD\_TRANSACTION
- ADD\_REPORT\_WRITER
- ADD\_OBN
- ADD\_INFOSET\_QUERY
- ADD\_FOLDER
- ADD\_BI\_ENTERPRISE\_REPORT
- ADD\_BI\_QUERY
- ADD\_BI\_TEMPLATE
- ADD\_KM\_DOCUMENT
- ADD\_PORTAL\_PAGE
- ADD\_VISUAL\_COMPOSER
- ADD\_WEBDYNPRO\_ABAP
- ADD\_WEBDYNPRO\_JAVA

| <b>REMOVE: Removes an application from a launchpad.</b> |                  |                  |             |                      |
|---------------------------------------------------------|------------------|------------------|-------------|----------------------|
| <b>Parameters</b>                                       | <b>Direction</b> | <b>Type kind</b> | <b>Type</b> | <b>Description</b>   |
| ID_APPLICATION_ID                                       | importing        | Type             | STRING      | GUID of Application. |



## Integration: Navigation in the Event Loop

If you call the `IF_FPM_NAVIGATION` method `NAVIGATE`, a new event object of type `cl_fpm_navigation_event` is created. This event object contains all the application parameters. The interface `IF_FPM_UI_BUILDING_BLOCK` contains the `PROCESS_EVENT` method, which allows you to call the navigation event and change these parameters.

To do this, implement the following code in the `PROCESS_EVENT` method:



### Syntax

```

1.  "First check if the event is a navigation event"
2.  check io_event->MV_EVENT_ID = io_event->gc_event_navigate.
3.  "Make a cast from the event object to the
    cl_fpm_navigation_event object"
4.  DATA lr_event type ref to cl_fpm_navigation_event.
5.  lr_event ?= io_event.
6.  "Get the business parameter"
7.  lr_bus_parameter ?= lr_event->mo_event_data.
8.  "Get the launcher parameter"
9.  lr_launcher_parameter ?= lr_event->mo_launcher_data.

```

Note the use of the following `lr_parameter` methods:

- `to_lpparam`: provides you with an internal table with the parameters
- `get_value`, `set_value` or `delete_value`: allow you to change a parameter



### Note

If the event processing requires further user interaction (for example, requesting further data via a dialog box), the event processing can be deferred by returning `EV_RETURN = IF_FPM_CONSTANTS~GC_EVENT_RESULT-DEFER`.

If the result of the event processing is ok, you can return `EV_RETURN = IF_FPM_CONSTANTS~GC_EVENT_RESULT-OK`; if the result of the event processing is not ok, you can return `EV_RETURN = IF_FPM_CONSTANTS~GC_EVENT_RESULT-FAILED`



### Note

To prevent a loss of data, you can implement the `NEEDS_CONFIRMATION` method. This method is located in the interface `IF_FPM_UI_BUILDING_BLOCK`. This method contains the navigation event and you can decide whether to raise a data-loss dialog box. To do this, you must return the following value: `eo_confirmation_request = cl_fpm_confirmation_request=>go_data_loss`.



## IF\_FPM\_NAVIGATE\_TO API

This interface provides you with a set of methods to launch an application without using a launchpad.

To access this Navigation API, use the interface `IF_FPM`. This provides the method `GET_NAVIGATE_TO( )` which returns an instance of the Navigation API `IF_FPM_NAVIGATE_TO`.

### Methods

This interface contains the methods described in the table below and the following bullet points:

| <b>LAUNCH_BEX_ANALYZER: Launches an application of type BEx Analyzer.</b> |                  |                  |                                  |                                                                               |
|---------------------------------------------------------------------------|------------------|------------------|----------------------------------|-------------------------------------------------------------------------------|
| <b>Parameters:</b>                                                        | <b>Direction</b> | <b>Type kind</b> | <b>Type</b>                      | <b>Description</b>                                                            |
| <code>IS_BEX_ANALYZER_FIELDS</code>                                       | importing        | Type             | <code>FPM_S_BEX_ANALYZER</code>  | Structure that contains the fields to add with BEx Analyzer application type. |
| <code>ET_MESSAGES</code>                                                  | exporting        | Type             | <code>FPM_T_T100_MESSAGES</code> | Error messages                                                                |
| <code>EV_ERROR</code>                                                     | exporting        | Type             | <code>BOOLE_D</code>             | Status: false - the application was added; true - an error occurred           |

The following are other methods with a similar interface to `LAUNCH_BEX_ANALYZER`, which allow you to launch a specified application:

- `LAUNCH_URL`
- `LAUNCH_TRANSACTION`
- `LAUNCH_REPORT_WRITER`
- `LAUNCH_OBN`
- `LAUNCH_INFOSET_QUERY`
- `LAUNCH_FOLDER`
- `LAUNCH_BI_ENTERPRISE_REPORT`
- `LAUNCH_BI_QUERY`

- LAUNCH\_BI\_TEMPLATE
- LAUNCH\_KM\_DOCUMENT
- LAUNCH\_PORTAL\_PAGE
- LAUNCH\_VISUAL\_COMPOSER
- LAUNCH\_WEBDYNPRO\_ABAP
- LAUNCH\_WEBDYNPRO\_JAVA



## Suspend and Resume

The Suspend and Resume feature enables an FPM application to remain in a suspended state when a user navigates to a URL. When the user navigates back to the FPM application, the Suspend and Resume feature allows the application to be resumed in the exact state it was before navigation occurred.

The basic settings to utilize this feature include the time out of suspended applications. Session Management and the Suspend and Resume feature are provided by technology layers like Web Dynpro ABAP Foundation, Portal, ABAP Server etc and are not provided or influenced by FPM. Suspend and Resume is supported in the following client environments:

- Stand-alone
- NWBC
- Portal



### Note

Suspend and Resume is currently limited to URL navigation. In the Report *Launchpad Customizing*, *Suspend and Resume* is only available for the URL application category of Report Launchpads. The same is also applicable to the API, in that only dynamic navigation to URLs via APIs can utilize the Suspend and Resume methods.

There is a uniform method to enable both Suspend and Resume across all the clients. But the method in which the external URLs get the information to navigate back to the Web Dynpro application varies. Only the FPM's methods to suspend and resume are detailed here.

With the Suspend and Resume feature, it is possible to pass parameters back and forth to the URL from the FPM application.

## Procedure

### Suspending via Static Launchpad Customizing



1. Open the Launchpad Customizing (transaction LPD\_CUST).
2. On the *Overview of Launchpads* screen, choose *New Launchpad*.
3. Enter the *Role*, *Instance* and *Description*. Choose *Continue*.
4. On the *Change Launchpad Role* screen, choose *New Application*.
5. Enter the following details:
  - *Linktext* – for example FPM\_TEST
  - *Application Category* – choose *URL*
  - *Application Parameters* - enter the URL of the application to be opened on suspension of the FPM application.

Note that you can also enter a description and application alias. The application alias is recommended if you use APIs of the launchpad.

6. Check the *Activate Suspend and Resume Functionality* checkbox.

When the user uses this launchpad application to navigate away from the FPM application, the FPM application is suspended.

#### **Suspending via Launchpad API**

It is possible from EhP1 of NW onwards to also use navigation dynamically, that is without creating a launchpad Customizing. It is possible to enable Suspend and Resume for such navigation too.

For information on how to get a handle to `IF_FPM_NAVIGATE_TO`, see [Navigation](#).

Once a handle is obtained to the `IF_FPM_NAVIGATE_TO` object, you can call the method `LAUNCH_URL` to open external applications. This method takes in an input parameter `IS_URL_FIELDS` of type `FPM_S_LAUNCH_URL`. In the structure `FPM_S_LAUNCH_URL`, the field `USE_SUSPEND_RESUME` must be set to `abap_true` or 'X'. When the application is launched (refer to Dynamic APIs of the launchpad), the FPM application is suspended.

#### **Resuming a Suspended Application**

When the user wants to navigate from the external URL back to the suspended FPM application, the FPM event loop is triggered. This is the entry point back into the application.

The application reacts to the FPM event `FPM_RESUME`, which is accessed via the constant `CL_FPM_EVENT=> GC_EVENT_RESUME`. The event data will contain the URL parameters that are passed from the external URL back into the FPM application.

The key to access this is via the following key parameter:

`CL_FPM_SUSPEND_RESUME_UTILITY=>CO_RESUME_URL_PARAMETERS`. The value obtained is an internal table of the type `TIHTTPNVP`, containing the URL key-values pair passed by the external application. Note that this data is available only during the

lifetime of the event object and is not stored by FPM. The application maintains a copy if the user needs to access this information later.

Sample code to resume an application is shown below (in the Component Controller's `PROCESS_EVENT` method):



### Syntax

```
1.  METHOD PROCESS_EVENT .
2.  "We will need to check the Navigation mode and set it to
   the launch pad accordingly.
3.  DATA lr_event          TYPE REF TO cl_fpm_navigation_event.
4.  "Check if this is the resume event.
5.  CASE io_event->mv_event_id.
6.      WHEN cl_fpm_event=>gc_event_resume.
7.          get_resume_parameters( io_event ).
8.  ENDCASE.
9.  Method GET_RESUME_PARAMETERS
10. DATA: lr_fpm_event_data TYPE REF TO if_fpm_parameter.
11. DATA: it_url_parameters TYPE tihttpnvp.
12. lr_fpm_event_data = io_event->mo_event_data.
13. CALL METHOD lr_fpm_event_data->get_value
14.     EXPORTING
15.         iv_key =
   cl_fpm_suspend_resume_utility=>co_resume_url_parameters
16.     IMPORTING
17.         ev_value = it_url_parameters.
```

At the end of this code, the internal table `it_url_parameters` contains the URL parameters passed back from the external application. The above mentioned code, along with other information, can be found in the test application `FPM_TEST_SUSPEND_RESUME` in the `APB_FPM_TEST` package.



## Handling Dialog Boxes

Depending on the action required, you can manage dialog boxes in the following ways:

- Using the `NEEDS_CONFIRMATION` method during the FPM Event Loop
- Using the `PROCESS_EVENT` method for the handling of application-specific dialog boxes
- Using the work-protect mode offered by the Portal and the NWBC (using the `IF_FPM_WORK_PROTECTION` interface)



## Triggering a Data-Loss Dialog Box in the FPM Event Loop

Each UIBB can request a data-loss dialog box during the FPM event loop.

To do this, return the pre-defined instance of the class `CL_FPM_CONFIRMATION_REQUEST` as detailed below:



#### Syntax

```
1. METHOD needs_confirmation
2.     IF ...
3.         eo_confirmation_request =
           cl_fpm_confirmation_request=>go_data_loss
4.     ENDIF
5. ENDMETHOD
```

To display other confirmation dialog boxes, create your own instance of the class `CL_FPM_CONFIRMATION_REQUEST` and add your own application-specific text.



## Handling Application-Specific Dialog Boxes

To process an event in method `IF_FPM_UI_BUILDING_BLOCK~PROCESS_EVENT` (see chapter FPM Events), it may be necessary to gather additional information from the user by means of a dialog box. Dialog boxes may contain simple text and buttons, but they may also be more complex and include input fields, checkboxes, etc.

The processing of dialog boxes in Web Dynpro programming can be cumbersome, since Web Dynpro dialog boxes cannot be processed in a synchronous way (i.e. trigger the dialog box, wait for it to be closed and continue processing). This means that the UIBB would need to return the result of the event processing (OK or FAILED) before the dialog box could be processed.

To achieve synchronous dialog box handling, the FPM allows you to defer the processing of the event loop and resume it after the dialog box has been processed. This procedure is described below:

### Procedure

#### Deferring current event processing

You defer the processing of the current event in the method `PROCESS_EVENT`. Sample code for this is shown below:



#### Syntax

```
1. ev_result = if_fpm_constants=>gc_event_result-defer.
```

#### Registering a dialog box

This procedure is purely Web Dynpro ABAP and not a feature of the FPM. Therefore, we recommend that you read the Web Dynpro ABAP documentation regarding Web Dynpro ABAP dialog boxes in general. Nevertheless, a short description of how to register a dialog box is detailed below:

Firstly, the registration of the dialog box with Web Dynpro needs to be triggered in the method `PROCESS_EVENT`, as this is the last method until program control returns to the FPM.

However, for an application-specific dialog box you need your own Web Dynpro ABAP View and the registration of the dialog box is only possible from within this View. For this reason, in the method `PROCESS_EVENT` you need to call a method of the View that is used for the application-specific dialog box. However, as View methods cannot be accessed from within methods of the component controller, you need to use the Web Dynpro ABAP event mechanism: raise an event in the method `PROCESS_EVENT` and register an event handler on the corresponding View.

The process for this is described below:

1. Create a new Web Dynpro ABAP View and name it `DIALOG_BOX_CARRIER`.
2. In the *Component Controller*, create a new Web Dynpro ABAP Event and name it `REGISTER_DIALOG_BOX_EVENT`.
3. In the method `PROCESS_EVENT` raise the Web Dynpro ABAP Event `REGISTER_DIALOG_BOX_EVENT`.
4. In the View `DIALOG_BOX_CARRIER`, create a new method and name it `REGISTER_DIALOG_BOX` of method type event handler for the event `REGISTER_DIALOG_BOX_EVENT`.
5. In the method `REGISTER_DIALOG_BOX`, use the ABAP Window API to create a dialog box, register action handler methods to the buttons of the dialog box and register the dialog box for opening.
6. Create Web Dynpro ABAP actions and handler methods for the actions that arise from the dialog box; in this case, from the *Yes* and *No* buttons. In the example above, the names are `ONRESUME_EVT_OK` and `ONRESUME_EVT_FAILED`.

The sample code below shows how this might look (the code uses a standard dialog box with buttons *Yes* and *No*):



#### Syntax

```

1. DATA: lo_api                TYPE REF TO if_Web
      Dynpro_component,
2.      lo_window_manager      TYPE REF TO if_Web
      Dynpro_window_manager,
3.      lo_view_api           TYPE REF TO if_Web
      Dynpro_view_controller,
4.      lo_dialog_box         TYPE REF TO if_Web Dynpro_window,
5. lo_api                    = Web Dynpro_comp_controller->Web
      Dynpro_get_api( ).
6. lo_window_manager        = lo_api->get_window_manager( ).
7. lo_view_api              = Web Dynpro_this->Web Dynpro_get_api( ).
8. lo_dialog_box            = lo_window_manager->create_dialog
      box_to_confirm(
9.     text                  = 'some dialog box text... '
10.    button_kind           = if_Web Dynpro_window=>co_buttons_yesno
11.    message_type         = if_Web
      Dynpro_window=>co_msg_type_question

```

```

12.     window_title      = 'some dialog box title...'
13.     window_position = if_Web Dynpro_window=>co_center ).
14. CALL METHOD lo_dialog box->subscribe_to_button_event
15.     EXPORTING
16.     button           = if_Web Dynpro_window=>co_button_yes
17.     action_name     = 'ONRESUME_EVT_OK'
18.     action_view     = lo_view_api.
19. CALL METHOD lo_dialog box->subscribe_to_button_event
20.     EXPORTING
21.     button           = if_Web Dynpro_window=>co_button_no
22.     action_name     = 'ONRESUME_EVT_FAILED'
23.     action_view     = lo_view_api.
24. lo_dialog box->open( ).

```

### Resuming the event

Once the required user input has been obtained, the frozen FPM event is continued (either receiving the result OK or FAILED). To do this, call the FPM method `RESUME_EVENT_PROCESSING` within the action handler methods for the buttons of the dialog box. The sample code below shows how this might look:



#### Syntax

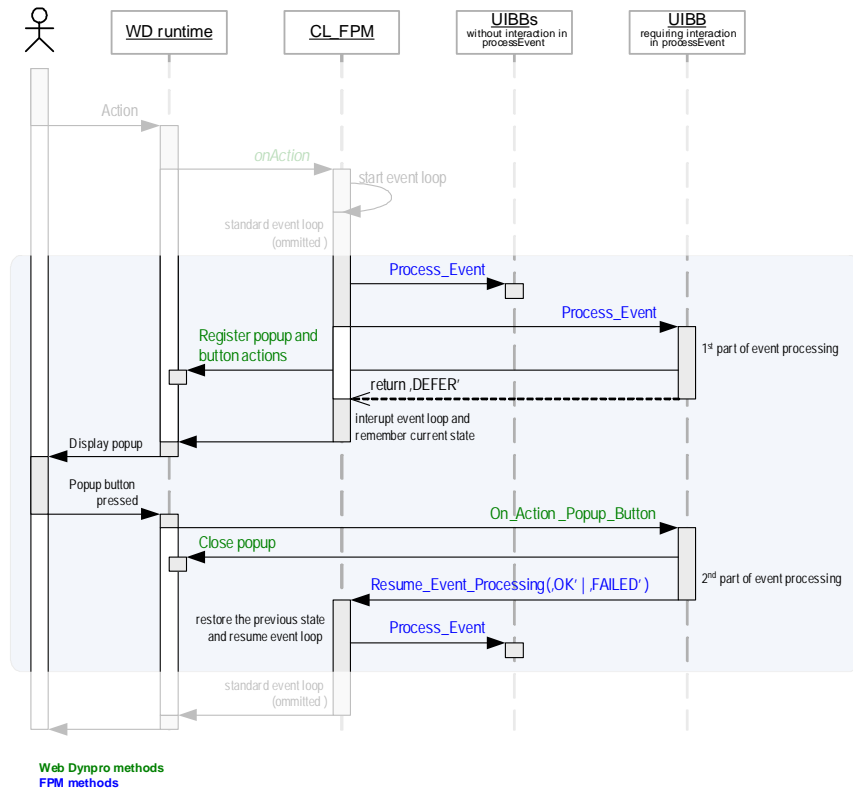
```

1. DATA lo_fpm TYPE REF TO if_fpm.
2. lo_fpm = cl_fpm_factory=>get_instance( ).
3. lo_fpm->resume_event_processing(
   if_fpm_constants=>gc_event_result-ok ).

```

After the event is resumed, the remaining UIBBs are processed (if there is more than one UIBB).

The figure below summarizes the behavior described above :



## IF\_FPM\_WORK\_PROTECTION Interface

The FPM allows the application to make use of the “work-protect mode” offered by the Portal and the NWBC (that is, to display a data-loss dialog box when the user closes the application without first saving the data).

To achieve this, the application must ‘tell’ the FPM whether it contains unsaved (“dirty”) data. For this, the FPM provides the Web Dynpro Interface IF\_FPM\_WORK\_PROTECTION. It contains only one method, which is described in the table below:

### METHODS

| Method Name | Method Description                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IS_DIRTY    | This interface can be implemented by any Web Dynpro component in your application which is known to the FPM (e.g. any UIBB or a shared-data component). At runtime, the FPM will detect all components implementing this interface. If any of these components signals unsaved data, then the application is marked as ‘dirty’. This application ‘dirty-state’ is then passed on by the FPM to the shell (i.e. the portal or the NWBC). |

 Note

The shell-API requires this information as soon as the application state changes. Therefore, the `IF_FPM_WORK_PROTECTION~IS_DIRTY` method is called by the FPM runtime during each roundtrip. Therefore, it needs to perform this very quickly. Note that the FPM does not necessarily call the method `IS_DIRTY` on all UIBBs that are currently visible. As soon as one UIBB informs the FPM that it has unsaved data, the FPM does not need to call the method on the remaining visible UIBBs. For this reason, do not assume that the `IS_DIRTY` method is called by the FPM on all visible UIBBs.

Your application can use the sample code shown below:



#### Syntax

```
1. METHOD is_dirty.  
2.     if * component contains unsaved data  
3.         ev_dirty = abap_true.  
4.     else.  
5.         ev_dirty = abap_false.  
6.     endif.  
7. ENDMETHOD.
```



## FPM Message Management

FPM message management is an integral part of FPM and is available to all applications that use the standard floorplans. It guarantees consistent and guideline-compliant message handling.

### Integration

### Prerequisites

### Features

FPM message management consists of two parts:

- *IF\_FPM\_MESSAGE\_MANAGER* Interface (*Message Manager*)

This interface provides you with methods to perform the following tasks:

- Clear messages
- Raise Exceptions
- Report messages

- *Message Region*

All messages to be reported are displayed in the *Message Region*. This UI element is included in all FPM applications.

You can make the following changes to the Message Region in the *Global Settings* dialog box:

- *Set the maximum message size*

When the application displays your messages, the message area expands to accommodate the number of messages that you enter in the *Maximum Message Size* field. Once the number of messages exceeds the maximum limit, a scroll bar appears in the message area. Thus you can view messages other than those immediately visible in the message area.

- *Turn on the message log*

You can produce a log of the messages for your application. When the message log is turned on, all the previously reported messages can be seen. When a message is to be reported, the *Display Message Log* link appears in the *Message Region*. Note that this link appears only when there is at least one message in the log.



Note

Note: You can also turn on the message log by using the URL parameter `FPM_SHOW_MESSAGE_LOG=X`. However, if you turn on the message log in the *Global Settings* dialog box, you cannot turn it off using the URL parameter.

## Activities

### More Information



### Using the FPM Message Manager

#### Procedure

1. In the Component Controller of your Web Dynpro Component, choose the *Attributes* tab.
2. Declare an attribute of the component globally (e.g. `MR_MESSAGE_MANAGER`) and declare the Associated Type as type `IF_FPM_MESSAGE_MANAGER`.
3. Choose the *Attributes* tab of your Component Controller. In the Web Dynpro `DOINIT` method, create a handle to the FPM Message Manager (which is a read-only attribute in the `IF_FPM` interface), as detailed in the code below:



#### Syntax

- ```

1. Method Web Dynpro DOINIT
2.     "Get the handle to the IF_FPM interface
3.     Web Dynpro_this->MR_FPM = CL_FPM_FACTORY=>GET_INSTANCE( )
4.     Web Dynpro_this->MR_MESSAGE_MANAGER = Web Dynpro_this-
    >MR_FPM->MO_MESSAGE_MANAGER
5. Endmethod

```

### Example



T100 based message. This example is taken from the demo applications and can be found in the Web Dynpro component FPM\_HELLOSFLIGHT\_OIF\_DEMO in the APB\_FPM\_DEMO package.



### Syntax

```
1. CALL METHOD Web Dynpro_THIS->MR_MESSAGE_MANAGER-
   >REPORT_T100_MESSAGE
2.     EXPORTING
3.         IV_MSGID                = 'APB_FPM_DEMO'
4.         IV_MSGNO                = 009
5.         IO_COMPONENT            = Web Dynpro_this
6.         IV_SEVERITY             =
   if_fpm_message_manager=>GC_SEVERITY_ERROR
7.         IV_LIFETIME             =
   if_fpm_message_manager=>GC_LIFE_VISIBILITY_AUTOMATIC
8.         IV_PARAMETER_1         = lv_carrid_string
9.         IO_ELEMENT              = lo_el_sflight_selection
10.        IV_ATTRIBUTE_NAME      = `CARRID`.
```

The T100 message is shown below:

*INSERT SCREENSHOT HERE*

When the message appears in the Message Region, the parameter &1 is replaced by the actual flight name.



### IF\_FPM\_MESSAGE\_MANAGER Interface

This programming interface provides you with methods for controlling message management in your FPM application in a logical manner.

It provides you with methods to perform the following tasks:

- Reporting messages

There are 3 methods available to report messages (including T100 and Bapiret2 messages).

- Raising exceptions

There are 4 methods available to raise exception messages (including T100 and Bapiret2 messages).

- Clearing messages

There are 1 method available to clear all messages.



### Methods for Reporting Messages

The methods for reporting messages are provided by the `IF_FPM_MESSAGE_MANAGER` interface. This interface provides the following methods for reporting messages:

- `REPORT_MESSAGE`
- `REPORT_BAPIRET2_MESSAGE`
- `REPORT_T100_MESSAGE`

Note the following information relating to all reporting methods:

- By default, the message is not mapped to a context element.
- *If there are minor inconsistencies while reporting the message, FPM automatically takes alternative action (unless an exception is raised). The following is the alternative action that FPM takes: If the message is reported to be bound to a context element and if the element or the attribute is missing, FPM reports the message without the binding.*
- FPM raises an exception in the following cases:
  - If the message lifetime is marked to be bound to a controller, but the controller is NULL or not reachable.
  - If the component for the message is missing.
  - If the Message Lifetime is set to Manual and View, but the element or attribute is missing.

#### Attributes

The attributes of the three methods for reporting messages are described in the table below.

Parameter	Relevant Method	Description
<code>IO_COMPONENT</code>	All	<i>Passes an object reference to the message manager. This object reference is used to store the message. Preferably, the Web Dynpro component, which raises the message, must be passed here. You can pass another object reference only in the event of exceptions where the object raising the message does not have a handle to the Web Dynpro component (e.g. an ABAP OO class) This is important for those messages whose lifetime is maintained manually by the application. (see <code>IV_LIFETIME</code>). When you create a message whose lifetime is manual, the application creating such a message must then delete the message once it is no longer needed. In this case, you must pass the component whose messages need to be</i>

Parameter	Relevant Method	Description
		<p>cleared. This helps to prevent messages from a different component being cleared by a component that has not raised them. This could happen when you re-use components from different areas</p>
IV_SEVERITY	All	<p><i>The severity of the message to be reported.</i> There are three possible values, as follows:</p> <ul style="list-style-type: none"> <li>• Error (E)</li> <li>• Warning (W)</li> <li>• Success (I)</li> </ul> <p>The default value is Error. These messages affect the navigation in different ways for each floorplan. Thus, navigation relating to an error message in a GAF application, may be different to navigation relating to an error message in an OIF application. The following three values can be passed:</p> <ul style="list-style-type: none"> <li>• GC_SEVERITY_ERROR for Error</li> <li>• GC_SEVERITY_WARNING for Warning</li> <li>• GC_SEVERITY_SUCCESS for Success</li> </ul> <p>This is an optional parameter. The default is Error.</p>
IV_LIFETIME	All	<p><i>Determines when, where and how long a message appears for.</i> This is a very important parameter and must be given special attention. This parameter is a combination of the following two elements:</p> <ul style="list-style-type: none"> <li>• <i>Lifetime:</i> Determines how long the message exists in the message area, i.e. the creation and deletion of the message. The available lifetimes are: <ul style="list-style-type: none"> <li>○ <i>automatic:</i> FPM handles the destruction of the message as defined by the UI guidelines for the floorplan</li> </ul> </li> </ul>

Parameter	Relevant Method	Description
		<ul style="list-style-type: none"> <li>○ <i>manual</i>: the application developer handles the deletion of the message from the message area.</li> <li>• <i>Visibility</i>: Determines when the message appears in the message area. The following values apply: <ul style="list-style-type: none"> <li>○ <i>Automatic</i>: FPM takes care of the visibility based on the UI guidelines</li> <li>○ <i>View</i>: the message is visible as long as the view to which the message is bound is available</li> <li>○ <i>Controller</i>: the message is visible as long as the controller that has raised the message is available (see the parameter controller IO_Controller for details)</li> <li>○ <i>Application</i>: the message is permanently displayed (until it is deleted manually by the application developer) and is visible whilst the application is running.</li> <li>○ <i>Pop-up</i>: the message is visible only in a dialog box.</li> </ul> </li> </ul> <p>The default values for both <i>Lifetime</i> and <i>Visibility</i> are <i>Automatic</i>. Not all combinations of lifetime and visibility are possible. Some combinations, e.g. <i>Lifetime = Manual + Visibility = Pop-up</i> are not available. The permitted combinations are as follows (showing the constant to be used - Lifetime + Visibility):</p> <ul style="list-style-type: none"> <li>• GC_LIFE_VISIBILITY_AUTOMATIC: <i>Automatic + Automatic</i> (Fully handled by FPM)</li> <li>• GC_LIFE_VISIBILITY_AUT_DIALO</li> </ul>

Parameter	Relevant Method	Description
		<p>G_BOX: <i>Automatic + Pop-up</i> (Creation and destruction handled by FPM; visible as long as the dialog box is visible)</p> <ul style="list-style-type: none"> <li>• GC_LIFE_VISIBILITY_MANU_VIEW: <i>Manual + View</i> (Should be deleted by the application; visible until the view that created it is visible)</li> <li>• GC_LIFE_VISIBILITY_MANU_CONT: <i>Manual + Controller</i> (Should be deleted by the application; visible as long as the controller that created it is visible)</li> <li>• GC_LIFE_VISIBILITY_MANU_APPL: <i>Manual + Application</i> (Should be deleted by the application; visible as long as the application is running)</li> </ul>
IV_PARAMETERS	All	<p>A group of parameters of the type Web Dynpro R_NAME_VALUE_LIST that can be stored along with the message. This will be passed to the Web Dynpro message manager as is and will have no visualize changes to the message. Refer to the Web Dynpro message manager documentation for further details</p>
IR_MESSAGE_USER_DATA	All	<p>Additional data that can be stored along with the message. This does not influence the message visually. This parameter can be used by the application developers to provide error resolution mechanism. See the Web Dynpro help for further details.</p>
IV_MESSAGE_INDEX	All	<p>Numerical value indicating the order in which the message is to be displayed. If no value is passed (this is an optional parameter), the message appears in the order in which the Web Dynpro runtime chooses to display it. Messages are sorted for display, according to the following attributes:</p> <ol style="list-style-type: none"> <li>1. Error severity</li> </ol>

Parameter	Relevant Method	Description
		<ul style="list-style-type: none"> <li>2. Message index (parameter MSG_INDEX)</li> <li>3. Context element (if it exists)</li> <li>4. Context attribute (if it exists)</li> </ul>
IO_ELEMENT	All	<i>A reference to a context element to which the message is bound. The message is then clickable and the focus shifts to a UI element bound to this context element.</i>
IV_ATTRIBUTE_NAME	All	<i>The element attribute to which the message must be mapped. This parameter is used in conjunction with the IO_ELEMENT.</i>
IV_IS_VALIDATION_INDEPENDENT	All	<i>Defines whether a message, referring to a context attribute or a context element, influences the execution of a standard action. If the parameter's value is ABAP_FALSE (default value), the standard action is no longer executed after this message is created. However, if the parameter's value is ABAP_TRUE, the standard action is executed.</i>
IO_CONTROLLER	All	<i>Pass the reference to the controller whose lifetime will dictate the lifetime of the messages which have the lifetime set to the context.</i>
IS_NAVIGATION_ALLOWED	All	<i>Use this flag if you need to allow navigation, even on an 'E' message in the GAF. Relevant for GAF applications only.</i>
IV_VIEW	All	<i>The name of the view of the dialog box. The error is then restricted only to the dialog box. Otherwise there is a side effect in that the error message (if a non-automatic type) is also reported on the main screen when the dialog box is closed. Relevant only if the message manager is used in application-specific dialog boxes.</i>
IV_MESSAGE_TEXT	REPORT_MESSAGE	<i>Any free text that must be reported in the message area. When used with the UI element and attribute parameters, it becomes a clickable free text message.</i>

Parameter	Relevant Method	Description
IS_BAPIRET2	REPORT_BAPIRET2_MESSAGE	<i>The BAPIRET2 structure directly in the message. The severity of the message is automatically selected from the BAPIRET2 structure. The T100 message that is embedded in the BAPIRET2 structure is used to display the message text. Additionally, the lifetime, visibility and context mapping can be set along with the BAPIRET2 structure.</i>  <i>Note: If the BAPIRET2 structure contains a severity value of A, the message is converted into an exception.</i>
IV_MSGID	REPORT_T100_MESSAGE	<i>Used when reporting a T100 based message. Supply the parameter with the message class.</i>
IV_MSGNO	REPORT_T100_MESSAGE	<i>The message number of the message class specified by the IV_MSGID.</i>
IV_PARAMETER_1 IV_PARAMETER_2 IV_PARAMETER_3 IV_PARAMETER_4	REPORT_T100_MESSAGE	<i>Optional parameters for the message.</i>

#### Mandatory Parameters

The table below shows which parameters are mandatory for each method:

Method	Mandatory Parameters
REPORT_MESSAGE	Message text
REPORT_T100_MESSAGE	Message class and message number
REPORT_BAPIRET2_MESSAGE	BAPIRET2 structure



#### Methods for Raising Exception Messages

The RAISE\_EXCEPTION methods are provided by the IF\_FPM\_MESSAGE\_MANAGER interface. This interface provides the following methods for raising exceptions:

- RAISE\_EXCEPTION
- RAISE\_T100\_EXCEPTION

- RAISE\_CX\_ROOT\_EXCEPTION
- RAISE\_BAPIRET2\_EXCEPTION

All exceptions are logged into the system with the following details:

- the method that was used to raise the exception
- the text of the exception
- additional text (if used)

From *SP13* onwards, there is no recovery mechanism from the exceptions.

#### Attributes

The following table describes the attributes of the four RAISE\_EXCEPTION methods.

Parameter	Relevant Method	Description
IV_TEXT	RAISE_EXCEPTION	Optional text that can be passed while raising a simple exception. This text is logged and can later be used for analysis
IV_MSGID	RAISE_T100_EXCEPTION	Message class ID for the T100 message. Use this parameter to raise an exception whose text is based on the T100 message mechanism.
IV_MSGNO	RAISE_T100_EXCEPTION	Message number of the T100 message class.
IV_PARAMETER_1 IV_PARAMETER_2 IV_PARAMETER_3 IV_PARAMETER_4	RAISE_T100_EXCEPTION	Optional message parameters.
IO_EXCEPTION	RAISE_CX_ROOT_EXCEPTION	The exception class inheriting from CX_ROOT. This parameter is a mandatory parameter.
IV_ADDITIONAL_TEXT	RAISE_CX_ROOT_EXCEPTION	Additional text to be added while reporting an exception based on CX_ROOT.
IS_BAPIRET2	RAISE_BAPIRET2_EXCEPTION	The BAPIRET2 structure for raising an exception.





## Method for Clearing Messages

This method is provided by the `IF_FPM_MESSAGE_MANAGER` interface.

Note the following information relating to this method:

- The method clears messages from the Message Region and acts upon all those methods that have Lifetime set to Manual.
- This is the only method to selectively clear those messages with a Lifetime set to Manual from the Message Region.
- This method ensures that messages from a different component are not cleared accidentally.
- The defaults for the parameters contain a negative semantic with respect to the method name; if the method is called with defaults, all the messages are deleted.

### Attributes

The following table describes the attributes for the `CLEAR_MESSAGES` method.

Parameter	Relevant Method	Description
<code>IO_COMPONENT</code>	<code>CLEAR_MESSAGES</code>	The component in which messages were previously reported. Only those messages that were reported from this component will be cleared. If this contains object references other than components, then those object references will be used. This is a mandatory parameter.
<code>IV_EXCLUDE_ERROR</code>	<code>CLEAR_MESSAGES</code>	Pass true if error messages belonging to the component are not to be deleted. This is an optional parameter and the default is false. This means that all the error messages belonging to this component will be deleted unless this parameter contains a true value. Looking at the parameter's name, the parameter indicates that the default value (false) has to be overridden only if error messages are to be saved from being cleared and this parameter contains negative semantic with respect to the method name.
<code>IV_EXCLUDE_WARNINGS</code>	<code>CLEAR_MESSAGES</code>	Default is false. Override it with true, if warnings raised for the component are to be

Parameter	Relevant Method	Description
		saved.
IV_EXCLUDE_SUCCES S	CLEAR_MESSAGE S	Default is false. Override it with true, if success messages raised for the component are to be saved.

## FPM Message Manager FAQ

### **Can I use the Web Dynpro message manager along with the FPM message manager?**

Yes. However, you create and maintain your own reference of the Web Dynpro message manager. Messages that are reported directly into the Web Dynpro message manager will not be maintained by FPM after they are reported and the application must handle the message independent of the FPM lifetime and visibility functions. Exceptions logged directly into the Web Dynpro message manager are not logged under the `FPM_RUNTIME_MESSAGES` checkpoint group.

### **I want to use the FPM floorplan but I do not want to use the FPM message area. Can I do this?**

Yes. Use the Web Dynpro message area. However, FPM message manager functions such as automatic lifetime handling, consolidated dialog box display etc is not then available.

### **Should I create a message area to use the FPM message manager?**

No. If you are using a standard floorplan (e.g. OIF or GAF), the message area is a standard part of an FPM application's UI.

### **Can I change the position of the message area?**

No. If you create an additional message area, the messages are repeated in both message area views.

### **I reported a message mapped to a context. I see only the text and the message is not navigable. What is happening?**

The element and the attribute do not contain valid references. In such a case, FPM still displays the message but it is not navigable.

### **When I raise an exception, the screen dumps. When I examine the stack I see that the `IF_FPM_MESSAGE_MANAGER` is the point where the dump occurs. Why?**

As of SP13 of NW 7.00 and SP03 of NW7.10, there is limited support for exception handling for FPM applications. Features such as recovery mechanisms from exceptions, special exception screens, etc are not available. All `RAISE_XX_EXCEPTION` methods in FPM will log any exception raised from the method and then force a dump. In this manner, the applications are terminated.



## Handling of Transactions

Transactions can be handled in a systematic manner in FPM by implementing the Web Dynpro interface [IF\\_FPM\\_TRANSACTION](#). This interface guarantees you the following advantages:

- There is a logical sequence in which the interface methods are called.
- The transaction steps can be split up into the sequence in which they are supposed to be processed.
- There is a check – save – validate sequence that provides high transaction integrity.
- The check – save – fail – recover sequence provides the required robustness to the transaction.

### Using the transaction interface

1. In the *Web Dynpro ABAP Workbench*, select a component that will contain the business logic to be executed on a save event. This could be any component known to the FPM (including any UIBB or Shared Data component used by the UIBBs of your application).
2. In the preview, choose Implemented Interfaces and, in edit mode, add the `IF_FPM_TRANSACTION` interface.
3. Save your entry.
4. In the *Action* column, choose the *Reimplement* button and ensure that the icon in the *Implementation State* column turns green.
5. Activate your component. In the *Activation* dialog box, ensure that all elements are selected and choose *Save*. You have now implemented the Transaction interface and if you open the *Component Controller* component, you can see the methods associated with it on the *Methods* tab.
6. In any Save event, data needs to be saved to a database. This can be realized in the following ways:
  - Use a shared data component (see section on shared data for further details).
  - Use direct context binding.
  - Use an assistance class.

The decision to use any or a combination of the above methods is taken by the application developer.

### FAQ

- On what event will these methods be called? These methods will be invoked by the standard FPM SAVE event.
- Can I have the FPM call these methods on my own custom event? No. These methods are called as part of the standard FPM event loop and hence will not react to custom events.
- Can I have multiple components implement this interface in the same application? Yes. The FPM will call all the methods on all the implementing components. But our general recommendation is to use only one central component for transaction handling.



## IF\_FPM\_TRANSACTION Interface

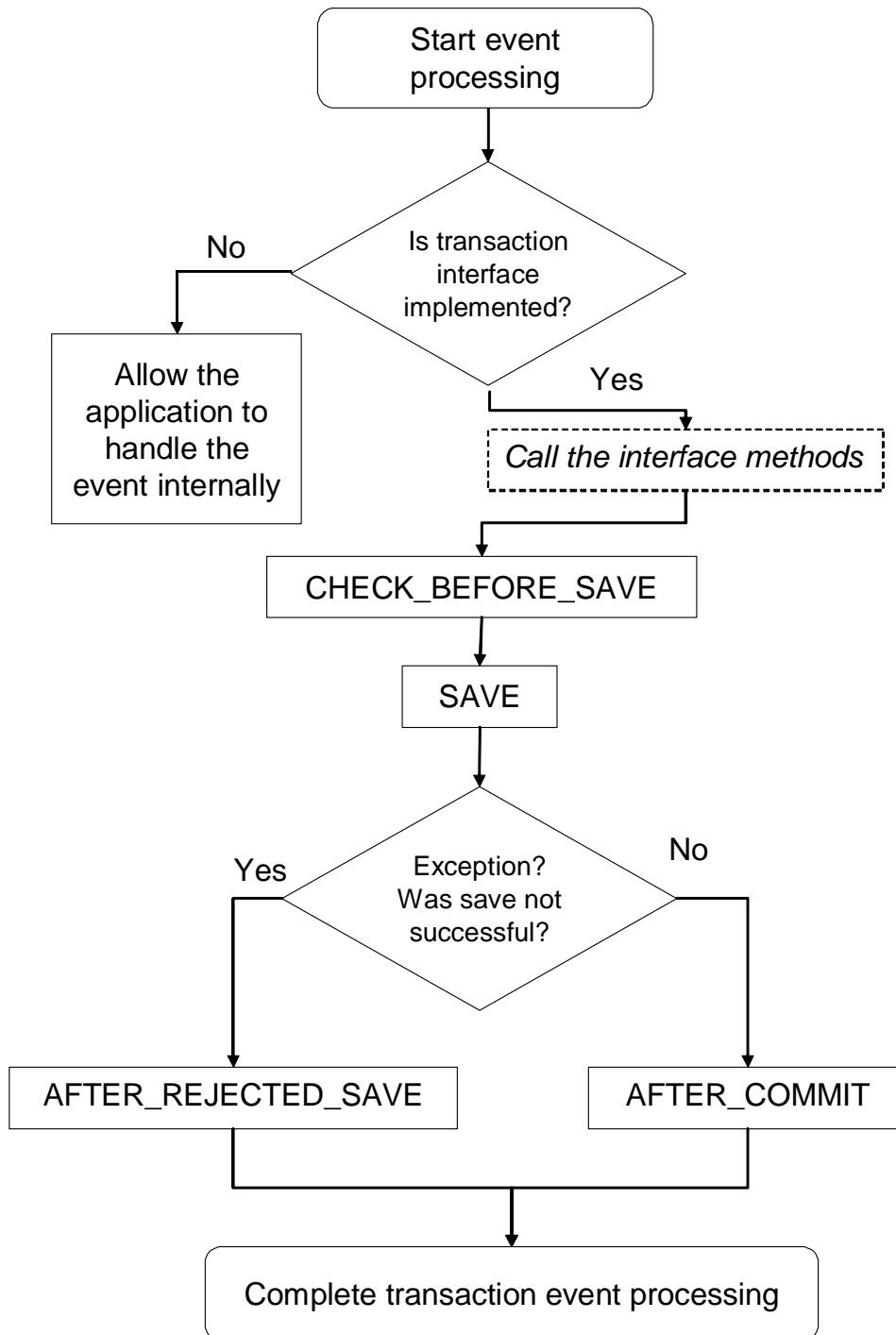
This Web Dynpro interface provides you with methods to handle transactions in a systematic manner by FPM. This is an optional interface; an application can handle the transactions independently, without implementing the interface.

### Methods

The IF\_FPM\_TRANSACTION interface contains the methods described in the table below. Note that once the interface is implemented, the FPM identifies the corresponding component, that has the method to be called, in sequence and calls the methods on this component in the same sequence as defined below.

Method Name	Method Description
CHECK_BEFORE_SAVE	This method has a return parameter which indicates whether the validation before a save to the database is successful. Use this method as the trial for save, and return a true if the trial save was successful and false if it was not.
SAVE	This method is used to perform the actual save and any possible commit. It is called when the CHECK_BEFORE_SAVE has returned a false (note that the semantic of the return parameter of the CHECK_BEFORE_SAVE is negative and reads 'rejected'. In other words, a false value for rejected means that the CHECK_BEFORE_SAVE was successful). If there are errors while saving, you must return rejected = true so that the AFTER_REJECTED_SAVE can be called. If the save was indeed successful, then the method AFTER_COMMIT is called. Refer to the flow chart for more details.
AFTER_COMMIT	You can perform clean up activities such as releasing database locks, releasing other resources, triggering an event for processing after a successful commit.
AFTER_REJECTED_SAVE	Here you can perform your roll back activities. You can also release locks and resources.

The methods are called in the sequence depicted in the figure below:



A detailed sequence diagram of the method calls can be found in the FPM Design Document.

## Resource Management

It is possible from NW EhP1 onwards for UIBBs to be made transient in their behavior. Transient behavior means that UIBBs, which are not visible, can be removed from memory so as to increase the performance and the memory footprint of the application.

### Releasing a Component

Technically, a UIBB is an interface view and this, by itself, cannot be released from memory, hence the FPM releases the component containing the UIBB based on certain rules. The rules are as follows:

- The application must use the new schema available from EhP1 onwards.
- The application developer must have set the transient flag to true via the *FPM Configuration Editor*.
- The FPM framework finds that it is technically feasible to release the component.
- The UIBBs implement the Resource Manager interface and do not veto the transient decision passed by the application via the `ON_HIDE` method.
- The UIBB has not implemented the Resource Manager interface (meaning that it does not have the possibility to veto).

A UIBB is defined and identified by the following key: *configuration + component + interface view*.

The transient behavior can be specified only during design time at the level of the application and not at the level of a UIBB or its usage.

The transient behavior of the UIBBS can be handled in one of the following ways:

- One UIBB per component

The component contains only one interface view which is used as a UIBB. When the UIBB is removed from the view assembly, the component that contains this UIBB is released.

- Multiple UIBBs per component

The component is released only when all the interface views that behave as UIBBs are no longer part of the view assembly, and the next set of UIBBs (for the forthcoming view assembly) does not contain a UIBB from this component. In such a case, the component is only released when all the interface views of this component are no longer part of the visible view assembly. Note that when one of the interface views (UIBBs) is removed from the view assembly, the component remains alive if other interface views of the same component are still part of the view assembly or part of the next view assembly.

If the application developer has set the global flag to transient, meaning that the UIBBs (components) can be released, then the FPM will investigate whether the component can be released.



## Note

There are instances when, even if the application developer has set the default to transient, the component containing the UIBBs cannot be released. These instances are described below:

- The component is held as a used component by another component.
- There are UIBBs from the same component that is still being displayed.
- The component implements an FPM interface that does not allow it to be released.

To evaluate whether to release a component, the FPM completes the following steps:

1. FPM checks for the presence of the *Global* flag in the *Global Settings* dialog box in the *FPM Configuration Editor*. If it is not present, then it will treat all the UIBBs for this application as non-transient and hence will not release any components.
2. FPM reads the configuration global flag to see if the configuration is set to transient. If the configuration is non-transient, then this information is passed on to the UIBBs and FPM ignores the transient behavior, i.e. it does not release the components.
3. FPM reads the configuration and sees that the global flag is set to transient. The following options are then available:
  - FPM checks if the `IF_FPM_RESOURCE_MANAGER` interface is implemented by the component. If so, it proceeds to the next step. Otherwise, it will check the technical feasibility of the component being released and, if it is feasible, it releases it. If it is not feasible, it retains the component.
  - FPM checks whether the component containing the UIBB under consideration is technically capable of being released (see the reasons bulleted above) and if so, it forwards the same status to the UIBBs via the `IF_FPM_RESOURCE_MANAGER`'S `ON_HIDE` method.
  - FPM checks that the interface is implemented and that, technically, it can be released. The same information is passed on to the UIBB. It checks for the veto value from the UIBB. If the UIBB has not vetoed the release state, then FPM releases it. Otherwise, it will retain it. In either case, the `ON_HIDE` method of the component is called.

### Settings for Transient Behaviour

Depending on how you want your application to use transient behavior, you can make the settings described in the following table.

<b>Requirement</b>	<b>Transient Flag</b>	<b>Implement IF_FPM_RESOURCE_MANAGER</b>	<b>Veto</b>	<b>Coding with the resource manager to handle application data</b>
I do not want transient behavior for any UIBB and I do not want to release any memory.	False	NA	NA	
I do not want transient behavior for any UIBB but I would like to release some resources.	False	NA	NA	Yes
I want all my UIBBs to be transient. I do not have the need to release any resources explicitly.	True	NA	NA	NA
I want all my UIBBs to be transient. I want to release some resources explicitly.	True	Yes	No	NA
I want only some of the UIBBs to be released. I would like to retain some due to business logic reasons.	True	Yes	Yes (only for those that do not need to be released).	Based on need from business logic.
I only want some of the UIBBs to be released. Some UIBBs I would like to retain due to business logic reasons. For those UIBBs that are transient, there is no	True	Only on those that need to veto (or not be released).	Yes (only for those that do not need to be released).	NA



Requirement	Transient Flag	Implement IF_FPM_RESOURCE_MANAGER	Veto	Coding with the resource manager to handle application data
need to release any resources.				

## Setting the Transient Flag

### Procedure

1. Start the *FPM Configuration Editor* for your application and go to the component configuration screen.
2. Choose *Change* and select *Global settings*.
3. The *Global Settings* dialog box contains the field for the transient setting.
4. Use the F4 help for *Transient State*. For the transient behavior, choose *T*.
5. Save the configuration.

### Result

All the UIBBs in the application are now transient.

## Using IF\_FPM\_RESOURCE\_MANAGER to Veto Release Decision

### Procedure

1. Open transaction SE80 and open the Web Dynpro component of your application.
2. Add IF\_FPM\_RESOURCE\_MANAGER to *Implemented Interfaces* tab.
3. In the Component Controller, on the *Methods* tab, the ON\_HIDE method is visible.

The ON\_HIDE method has an importing parameter called IV\_RELEASE\_COMPONENT, which provides information to the UIBB about the FPM's decision on the release feasibility for the component containing this UIBB. The UIBB reacts to this parameter only if the value is true. If the UIBB does not want itself to be released, then it sets the exporting parameter EV\_VETO\_RELEASE to true (the default is false).

FPM will use the veto parameters only if the `IV_RELEASE_COMPONENT` is true. If the UIBB sets the veto to true, then the component containing the UIBB is not released, even if it is capable of being released.

The sample code below demonstrates this:



#### Syntax

```

1.  method ON_HIDE .
2.      data: lv_veto type boole_d.
3.      IF IV_RELEASE_COMPONENT = abap_true.
4.          "do some business logic here and based on it, set the
         flag
5.          lv_veto = abap_true.
6.      ENDIF.
7.      IF lv_veto = abap_true." some bus.
8.          EV_VETO_RELEASE = abap_true. "This UIBB will not be
         released.
9.      ENDIF.
10. endmethod.

```

The following table is helpful in understanding the final action taken by FPM.

<b>IV_RELEASE_COMPONENT</b>	<b>EV_VETO_RELEASE</b>	<b>FPM action</b>
False	True/False	Ignore the veto value; do not release component.
True	False	Release the component.
True	False	Do not release the component.



### Using an FPM Application Controller

Sometimes it is necessary for the application to participate in all FPM events that happen during the entire lifetime of the application, with one arbitrary single component instance. This might be necessary for controlling and steering the application as a whole.

This is not possible, for example, with simple UIBBs since the methods provided by the Web Dynpro interface [IF\\_FPM\\_UI\\_BUILDING\\_BLOCK](#) only participate in the FPM event loop when the corresponding UIBBs are visible at the time the event loop happens or become visible after the current event loop has finished successfully. Furthermore the UIBBs cannot make assumptions about the sequence in which they are called. Therefore, an application controller is provided that closes this gap and provides the possibility to control and steer the application as a whole.

#### Implementing the Application Controller

The application controller is a singleton instance of a Web Dynpro component provided by the application. In order to use a Web Dynpro component as an application controller, complete the following steps:

1. Choose a Web Dynpro component and implement the Web Dynpro interface [IF\\_FPM\\_APP\\_CONTROLLER](#).
2. Insert the component you have chosen into the OIF or GAF component configuration.

To do this, open the component configuration with the *FPM Configuration Editor*. Choose *Display* and choose *Global Settings*. In the dialog box, enter the component.

Regarding the behavior of instantiating the Web Dynpro components and their participation within the FPM event loop, the Web Dynpro interfaces provided by the FPM can be divided into two categories:



Note

When using the interfaces `IF_FPM_APP_CONTROLLER` and [IF\\_FPM\\_OIF\\_CONF\\_EXIT](#) (or `IF_FPM_GAF_CONF_EXIT`) together, they must be implemented by the same Web Dynpro component. Furthermore, it is recommended to implement the Web Dynpro interface [IF\\_FPM\\_SHARED\\_DATA](#) also in that Web Dynpro component (but only if this Web Dynpro interface is needed).



## **IF\_FPM\_APP\_CONTROLLER Interface**

This Web Dynpro interface provides you with methods to allow the application to participate in all FPM events that happen during the entire lifetime of the application.

### **Methods**

This interface contains similar methods to the Web Dynpro interface `IF_FPM_UI_BUILDING_BLOCK`.

The interface `IF_FPM_APP_CONTROLLER` has two corresponding methods with the prefix `BEFORE_` and `AFTER_` for each of the `IF_FPM_UI_BUILDING_BLOCK` methods, for example, `BEFORE_PROCESS_EVENT` and `AFTER_PROCESS_EVENT`.

As the names suggest, the method `BEFORE_PROCESS_EVENT` is called immediately before another call to the corresponding UIBB method `PROCESS_EVENT`; the `AFTER_PROCESS_EVENT` is called immediately after all calls to `PROCESS_EVENT` are finished.



## **Using an Application-Specific Configuration Controller**

Using an application-specific configuration controller (AppCC) allows you to do the following:

- Make global checks (checks affecting more than one UIBB)
- Make global adjustments for FPM events
- Read the structure of your application at runtime
- Change the structure of your application dynamically


This is the place where all actions affecting more than one single UIBB can be performed. Using an AppCC is optional; implement an AppCC only if you need one of the features which the AppCC offers.

### Implementing an AppCC Component

To provide your application with an AppCC, you implement one of the following Web Dynpro interfaces in a Web Dynpro component:

- `IF_FPM_OIF_CONF_EXIT` for an OIF application
- `IF_FPM_GAF_CONF_EXIT` for a GAF application

This Web Dynpro component is either one of the components already used within your application or is a completely new one. To declare the AppCC component to FPM, proceed as follows:

1. Start the *FPM Configuration Editor* for your application component and open the *Component Configuration* screen.
2. In the control region, choose **Change** → *Global Settings* .
3. In the *Global Settings* dialog box, enter the *Web Dynpro Component* and the *Configuration Name*.
4. Choose Save.



#### Note

If your AppCC has declared a static usage to a component implementing `IF_FPM_SHARED_DATA`, this shared data component is instantiated and attached automatically by the FPM framework. This ensures that all components within your application, which access the shared data component, will see the same instance of it.

### Methods

The AppCC interface contains only one method for each floorplan application:

- `OVERRIDE_EVENT_OIF`
- `OVERRIDE_EVENT_GAF`

These methods pass an object of type `IF_FPM_OIF` (or `IF_FPM_GAF`), which serves as an API for the applications. The `OVERRIDE_EVENT_OIF` (or `OVERRIDE_EVENT_GAF`)

method is called at the start of event processing on all visible UIBBs immediately after the `FLUSH` method has been called.

## Features

The AppCC application programming interface provides you with the following features:

- *Canceling events*

With the AppCC you can perform global checks which apply to more than one UIBB. For checking purposes, the event is stored as an attribute in the `IF_FPM_OIF` (respectively `IF_FPM_GAF`) interface of the AppCC. You can cancel an event out of the AppCC by calling the `CANCEL_EVENT` method of the AppCC.

- *Selecting a variant*

If there is more than one variant configured, you can select a specific variant to be used in an event by calling `SET_VARIANT` method in the `IF_FPM_OIF` respectively `IF_FPM_GAF` interface.

- *Adjusting events*

The `IF_FPM_OIF` respectively the `IF_FPM_GAF` interface provides the currently processed FPM event as a changeable attribute. Therefore, it is possible to change an event by adding, removing, or changing event parameters. You also can replace an event.

As the AppCC is called right at the beginning of the event loop, changing an event has the same result as if changed event had been raised instead of the original event.

- *Reading the configuration at runtime*

The AppCC provides you with several methods which allow you to read the configuration data at runtime. The following table gives you an overview of all methods available for all types of floorplans.

<i>Method</i>	<i>Method Description</i>
<code>GET_CURRENT_STATE</code>	Returns the current navigation state within the application.
<code>GET_VARIANTS</code>	Returns a list of all available variants.
<code>GET_UIBB_KEYS</code>	Returns a list of all UIBB assigned to a specified main step, substep, main view or subview.
<code>GET_UIBB_KEYS_FOR_CONF_STEP</code>	Returns a list of all UIBB assigned to a confirmation screen.
<code>GET_UIBB_KEYS_FOR_INIT_SCREEN</code>	Returns a list of all UIBB assigned to an initial

N	screen.
---	---------

The following table gives you an overview of all methods available for an OIF application.

<i>Method</i>	<i>Method Description</i>
GET_MAINVIEWS	Returns a list of all main views for a given variant.
GET_SUBVIEWS	Returns a list of all subviews for a given main view.

The following table gives you an overview of all methods available for a GAF application.

<i>Method</i>	<i>Method Description</i>
GET_MAINSTEPS	Returns a list of all main steps for a given variant.
GET_SUBSTEP_VARIANTS	Returns a list of all substep variants for a given main step.
GET_SUBSTEPS	Returns a list of all substeps for a given substep variant.
GET_HIDDEN_MAINSTEPS	Returns a list of all hidden main steps for a given variant.

- *Changing the configuration at runtime*

The AppCC provides you with several methods if you want to change the configuration data at runtime. The following table gives you an overview of all methods available for all types of floorplans.

<i>Method</i>	<i>Method Description</i>
ADD_UIBB	Adds dynamically another UIBB to a main view, subview, main step, or substep.
REMOVE_UIBB	Removes dynamically another UIBB to a main view, subview, main step, or substep.

The following table gives you an overview of all methods available for an OIF application.

<i>Method</i>	<i>Method Description</i>
ADD_MAINVIEW	Adds dynamically another main view at runtime.
REMOVE_MAINVIEW	Deletes dynamically a given main view at runtime.
ADD_SUBVIEW	Adds dynamically another subview at runtime.
REMOVE_SUBVIEW	Deletes dynamically a given subview at runtime.

RENAME_MAINVIEW	Renames dynamically a given main view at runtime.
RENAME_SUBVIEW	Renames dynamically a given subview at runtime.
SET_SELECTED_SUBVIEW	Changes the target subview within a given main view. This method must only be used in order to enforce a given main view to switch to the provided subview instead of the default subview.

The following table gives you an overview of all methods available for a GAF application.

<i>Method</i>	<i>Method Description</i>
RENAME_MAINSTEP	Renames dynamically a given main step at runtime.
RENAME_SUBSTEP	Renames dynamically a given substep at runtime.
ENABLE_MAINSTEP	Enables or disables a given main step at runtime.
HIDE_MAINSTEP	Hides a given main step within the roadmap. The affected main step will not be visible as a main step in the roadmap anymore. Nevertheless, the hidden main steps continue to be processed in the background in order to keep the business logic untouched.



## Sharing Data between UIBBs from different Components

When the UIBBs of an application are implemented in several components, there is often the need to share data between these components. Technically, there are several approaches which you can take to achieve this. This is described in the following chapters.

For this purpose, the FPM offers Shared Data components.

This is an optional FPM feature which meets most applications' demands. However, if needed, it can be replaced by other technical alternatives as described in "Other Options for Sharing Data".

### Using a Shared Data Component

A shared data component is a Web Dynpro component which implements the `IF_FPM_SHARED_DATA` interface. This interface contains no methods or attributes but serves as a marker interface only. Each component (e.g. UIBB, `FPM_OIF|GAF_CONF_EXIT` component) which wants to use a shared data

component needs to declare a usage to the shared data component. For this, the technical type of the usage does not need to refer to `IF_FPM_SHARED_DATA` (this would mean that it would not have accessible methods/attributes) but link to the actual component itself. The lifecycle handling is now handled completely by the FPM. Whenever a component is instantiated by the FPM (e.g. a UIBB which is configured for a given screen), the FPM analyzes all usages of that component. If it detects a usage pointing to a component which implements the `IF_FPM_SHARED_DATA` interface, a singleton of this shared data component is automatically attached to the usage.

As a result, an application must proceed as follows to share data using the shared data interface:

- Create a component which implements the `IF_FPM_SHARED_DATA` interface.

This component contains methods to retrieve data from the business logic and exposes the extracted UI data via its Web Dynpro context or interface methods.

- Each component accessing this shared data defines a usage of the shared data component. This usage is automatically instantiated by the FPM.

The consuming component can now communicate with the shared data component via Web Dynpro context mapping, attribute access or method calls.

#### **Other Options for sharing Data**

There are other options to share data between Web Dynpro components besides the FPM shared data concept. There are occasions when it is best not to use a Shared Data component, as detailed below:

- There is already an application-specific API available which serves as a ‘data container’ and can be accessed by several components.
- The data needs to be shared not only between Web Dynpro components but also between other entities, such as ABAP OO classes, function groups, etc.
- The amount of data to be shared is so large that putting it into a Web Dynpro context would result in performance and memory consumption issues.

In these cases, the application can consider using techniques such as the following:

- An ABAP OO class which is accessible as a singleton, so that all consumers share the same instance.
- A function group with appropriate function modules.



#### **Embedding and FPM Application**

FPM was designed for building standalone applications. However, it is possible (with some restrictions) to embed an FPM application within another Web Dynpro application.



To do this, proceed as follows:

1. Create a usage for the component `FPM_OIF_COMPONENT` for OIF applications (or `FPM_GAF_COMPONENT` for GAF applications) within the embedding component.
2. Embed the `FPM_WINDOW` Interface View within one of the views of the embedding component.
3. Manually create the FPM component to be used (as you must provide the configuration key of the floorplan component). This is best done as soon as possible.

In most cases, this is the Web Dynpro `DOINIT` method of the embedding application's component controller, as the sample code below shows:



#### Syntax

```
1.  method Web DynproDOINIT .
2.      data: lo_usage type ref to if_Web Dynpro_component_usage,
3.            ls_conf_key type Web Dynpro_config_key.
4.      lo_usage = Web Dynpro_this->Web Dynpro_cpuse_fpm_usage( ).
5.      if lo_usage->has_active_component( ) = abap_true.
6.          lo_usage->delete_component( ).
7.      endif.
8.      ls_conf_key-config_id = "ID configuration of FPM
component".
9.      * recreate component using new configuration ID
10.     try.
11.         call method lo_usage->create_component
12.             EXPORTING
13.                 component_name    = 'FPM_OIF_COMPONENT'
14.                 configuration_id = ls_conf_key.
15.     catch cx_Web Dynpro_runtime_api .
16.     endtry.
```



#### Note

The following remarks relate to the above sample code:

- The configuration you pass is the configuration key of component `FPM_OIF_COMPONENT`. You cannot pass the application's configuration key.
- The code example names the usage `FPM_USAGE`. If you name it differently, adjust the following line: `lo_usage = Web Dynpro_this->Web Dynpro_cpuse_fpm_usage( )`.
- The example is for an OIF application; for a GAF application, replace `FPM_OIF_COMPONENT` by `FPM_GAF_COMPONENT`.
- The `delete_component( )` call is not necessary for simple static embedding. However, you need it if you want to change the embedded FPM application in the future.

## Procedure

### Constraints

- FPM allows only one instance running at the same time within one internal mode. Therefore, you cannot embed more than one FPM application at the same time. It is possible to switch the embedded FPM application, replacing one FPM application by another. You can assure this if you only use one Usage to an FPM component within your application. This forces you to delete the old FPM component before creating a new one.
- You can not embed an FPM application within another FPM application.
- You can not pass a configuration key for the IDR (header area). Therefore, the header appears without configuration settings; these you can set programmatically at runtime.
- You can not pass application parameters for the FPM application, as the application is now unknown to FPM.