|  |
| --- |
| Macmillan |
| Platform-X Menus |
| Specification of Platform-X Menu Structure |
|  |
| **David Newman** |
| **10/14/2011** |

|  |
| --- |
| This document contains a specification of the menu system for Platform-X. |

Contents

[Overview 3](#_Toc306603967)

[Platform-X Skeleton Structure 3](#_Toc306603968)

[PX\_MENUS 4](#_Toc306603969)

[PX\_MENUITEM\_TEMPLATES 4](#_Toc306603970)

[PX\_MENU\_EDITORS 4](#_Toc306603971)

[Menu Data 4](#_Toc306603972)

[Common Metadata 4](#_Toc306603973)

[allow 4](#_Toc306603974)

[bfw\_display\_flags 5](#_Toc306603975)

[bfw\_edit\_flags 5](#_Toc306603976)

[bfw\_menu\_callbacks 5](#_Toc306603977)

[Menu 6](#_Toc306603978)

[MenuItem 6](#_Toc306603979)

[Methods 7](#_Toc306603980)

[MenuEditorCollection 7](#_Toc306603981)

[MenuEditor 8](#_Toc306603982)

[Methods 9](#_Toc306603983)

[Page Layout Engine 9](#_Toc306603984)

[Business Layer 9](#_Toc306603985)

[IPageService 9](#_Toc306603986)

[Application Layer 10](#_Toc306603987)

[PageLayoutController 10](#_Toc306603988)

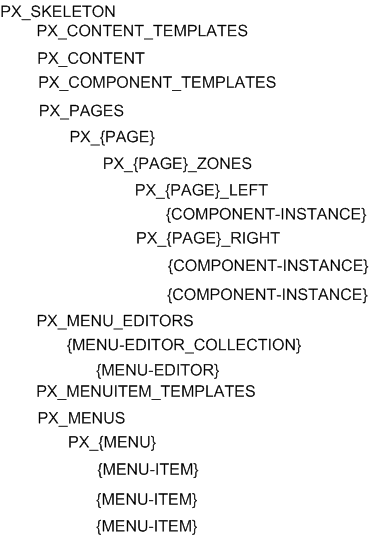
# Overview

Menus provide users the ability to navigate between pages and content within Platform-X. Menus are initially created in a product course by an editorial team, but may be modified and extended in derivative courses by instructors.

This document provides the details on how the menu construct is stored in DLAP as well as the code structure responsible for extracting the menu and rendering it to the screen. Workflow for both users and instructors is provided throughout covering topics including creating new menu items, editing existing menu items, and deleting menu items.

## Platform-X Skeleton Structure

The following image shows the Platform-X Skeleton structure into which all content and application settings data must be stored.



The nature of all parts of the skeleton structure are outside the scope of this document so only the sections necessary to the Menu system are discussed here.

### PX\_MENUS

The PX\_MENUS item is a folder that contains Menu items. Each Menu item contains one or more MenuItem items that constitute the menu.

### PX\_MENUITEM\_TEMPLATES

Some products have a base set of editorially controlled menu items. These menu items may be selected by the instructor for inclusion in derivative courses. The PX\_MENU\_ITEM\_TEMPLATES folder contains a MenuItem item to act as a template for adding the editorially created items to menus.

### PX\_MENU\_EDITORS

This folder contains items that represent collections of MenuItemEditor items. The contents of this folder are used to determine what types of custom menu items a user can add to the course's menu.

# Menu Data

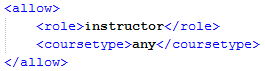
Like all content and settings in Platform-X, the data for the menu system are stored as XML based documents in DLAP. The following sections describe each type of item and how it is used by the menu system.

## Common Metadata

There are certain metadata that are used across the items used in the Menu system. This section describes them in a single place for the sake of efficiency.

### allow

The allow element restricts access to a Menu or MenuItem. The following listing shows an example allow element.



As can be seen in the above listing, the allow element has two child-elements. Role restricts access to users in the role specified, e.g. instructors will make it so that only instructors have access to the Menu or MenuItem. The coursetype child-element restricts access to a specific type of course, e.g. product makes it so that access is only granted on a product course.

The final decision of whether or not to allow access to a Menu or MenuItem is based on the AND of the current user's role and the type of course being accessed. Both the role and the course type must match the value of the allow element in order for access to be granted.

The allow element is not used on its own, but rather as a child element for other metadata. When there are multiple allow elements the effect is an ORing of all of the allow elements. Therefore if the restrictions of any allow element is met, the access will be granted.

### bfw\_display\_flags

The bfw\_display\_flags element contains child allow element which determine whether or not the Menu will be rendered for the user. The allow elements work as previously described and the current user's role as well as the course type being accessed must match the values specified.

### bfw\_edit\_flags

The bfw\_edit\_flags element contains child allow elements which determine whether or not the Menu or MenuItem can be edited by the user. The allow elements work as previously described and the current user's role as well as the course type being accessed must match the values specified.

The Page Layout Engine applies edit flags based on hierarchy. This means that even if a user is allowed to edit a Menu they may not be allowed to edit a specific MenuItem. In order to edit any item, the user must meet the constraints specified by the bfw\_edit\_flags element on the item itself as well as on the items higher in the hierarchy.

### bfw\_menu\_callbacks

The bfw\_menu\_callbacks elements contains "method" child-elements that describe the operations that can be performed by a MenuItem. Every "method" element has a "name" attribute that allows the Page Layout Engine to find the "method" necessary to support a specific action defined by the implicit interface for MenuItems.

Since the Page Layout Engine doesn't know how to invoke the compiled code for the MenuItem it uses several attributes to determine how to invoke the method. The "controller", "action", and "url" attributes are used by the Page Layout Engine to compute a URL to load in order to execute the method. If the destination is within the application then the "controller" and "action" attributes are used to compute the internal URL to the method. For off-site methods the "url" attribute is used. The "type" attribute is used to determine the style of display for the method's result. The value "fne" in the "type" attribute means that a Focused and Engaged pop-up will be used to display the content pointed to by the method. The default value of "navigate" means that the user will be taken to a different page inside or outside of the application.

Method elements can have "parameter" child elements. These elements will be used to construct the query string for the URL computed based on the "controller", "action", and "url" attributes. Some parameter values can be supplied by the Page Layout Engine. For example, when editing the MenuItem the MenuItem's ID is passed automatically.

## Menu

Menus are the core construct in the menu system. A Menu is a Folder that contains MenuItems. Menus provide users with the ability to navigate between pages and content in Platform-X. The following listing shows the XML definition of a typical Menu.



## MenuItem

MenuItem's are XML documents that describe a specific link in the Menu. A MenuItem's primary role is to tell the system where to take the user if that MenuItem is selected. Like Menus, MenuItems make use of the bfw\_display\_flags and bfw\_edit\_flags metadata elements to control access to the item.

The following listing shows an example XML document that represents a MenuItem.



The following sections describe any metadata elements that are specific to the MenuItem document type.

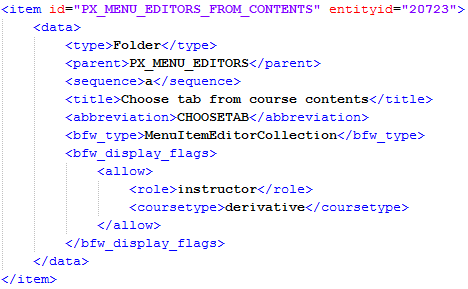
### Methods

#### Navigate

The navigate method is part of the implicit interface defined for MenuItems. The navigate method is responsible for rendering the content the user expects to see when the MenuItem is selected. There are no standard parameters defined for the navigate method.

## MenuEditorCollection

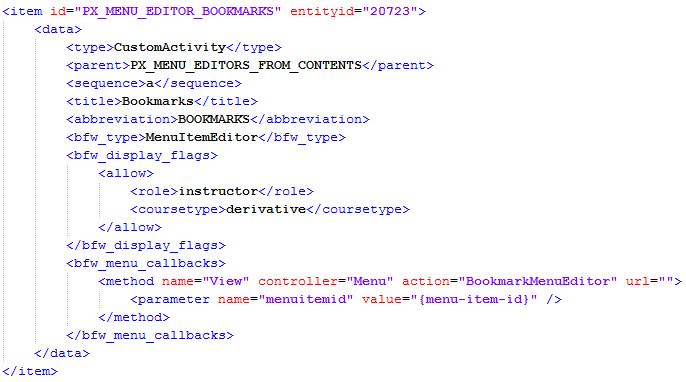
MenuEditorCollections are XML documents used to group MenuEditors. This grouping is used to drive specific sections of the UI when creating or editing MenuItems. In general the MenuEditorCollection is simply a folder that contains MenuEditor items, but it also uses the bfw\_display\_flags elements to control access.



### MenuEditor

MenuEditors are XML documents used to store metadata about a specific user-interface that can be used to create MenuItems. This allows the Page Layout Engine to support a variety of MenuItems, each with a custom UI that is optimized for creating and editing it.

The following listing shows an example MenuEditor XML document.



### Methods

#### View

The view method is invoked when the Page Layout Engine wants to allow the user to create or edit a MenuItem. When the particular type of item the user is interested in is selected, the appropriate editor is displayed. The view method is automatically passed a "menuitemid" parameter when it is editing as opposed to creating a new MenuItem.

# Page Layout Engine

The page layout engine is a set of server and client side components that handler loading, rendering, and general client side behaviors of pages, zones, widgets, and menus.

## Business Layer

The business layer is responsible for loading all necessary data from DLAP and turning it into object consumed by the application's controllers. The following service contract must be defined in the Bfw.PX.Biz.ServiceContracts project. The implementation for this contract should be placed in the Bfw.PX.Biz.Direct.Services project.

### IPageService

**Method:** PageDefinition LoadPageDefinition(string pageName)

**Description:** This method loads the Page item with the id of format PX\_{pageName} along with all of its Zones, and the Widgets in those Zones. Any Menus that are referenced by the Page item are also loaded at this time along with any items they may have. Properties of the IBusinessContext must be used to filter the list of Widget items in the Page's Zones based on each Widget's bfw\_display\_flags.

**Method:** void MoveWidget(string pageName, string zoneName, string widgetId, string sequence)

**Description:** This method moves the Widget instance so that it is in the specified Zone at the specified position.

**Method:** void SetWidgetDisplay(string widgetId, WidgetDisplayOptions display)

**Description:** This method changes the widget's bfw\_display\_flags values to match the display parameter. This can be used to hide the widget from view in matching cases.

**Method:** void RemoveWidget(string widgetId)

**Description:** This method removes the widget from any Zones it is currently in.

**Method:** Widget AddWidget(string pageName, string zoneName, string widgetTemplateId)

**Description:** This method will create a copy of the specified widget and put it in the correct page zone. The newly created Widget will be returned.

**Method**: MenuItem AddMenuItem(string menuId, string menuItemTemplateId)

**Description**: This method will create a copy of the specified menu item and put it in the correct menu. The newly created MenuItem will be returned.

**Method**: MenuItem SaveMenuitem(string menuId, MenuItem menuItem)

**Description**: This method will save existing MenuItems as well as create new custom MenuItems.

**Method**: void MoveMenuItem(string menuId, string menuItemId, string sequence)

**Description**: This method will move an existing MenuItem to a new position in the given menu.

## Application Layer

The application layer is responsible for loading data from the Business Layer and rendering it to a view.

### PageLayoutController

This controller is responsible for loading a PageDefinitionModel and rendering it to a view.

**Action:** ActionResult RenderPage(string pageName)

**Description:** This action method will load the PageDefinition from the IPageService and convert it to a PageDefinitionModel. The resulting model will be rendered by the RenderPage view.