Extension Installation Files: Version 2.1

An extension installation file is an XML file (with the extension .mxi) that provides the following information for the Extension Manager:

- Extension name and version number.
- Information about each file included in the extension.
- Information about how users access the extension from an Adobe application.
- Information about language-specific files for multilingual extensions.

This document describes the tags used in the installation file. For a list of each tag and compatible Adobe products, see "Tags and their compatible products" on page 43. After reading about these tags, you can examine the sample installation file in the Extension Manager's Samples folder, or you can make a copy of the blank installation file and fill in values for the attributes.

When you create your extension installation file, give it a filename that is valid on both Windows and Mac OS, is no more than 20 characters long, and contains no spaces.

This article contains the following sections:

About careful XML coding

XML files have fairly strict syntax requirements. When you're creating or editing an extension installation file, make sure you use correct XML syntax:

- Every attribute value must be enclosed in a single pair of double quotation marks. For example, `version = 1.0.0` and `version = ""1.0.0""` are both incorrect syntax; instead, use `version = "1.0.0"`.
- A tag defined as an empty tag (a tag with no contents) must end with `/>`. Do not include any spaces between the slash and the closing angle bracket.
- Each attribute name must be preceded by a space (or other form of white space). In particular, if you use more than one attribute in a tag, you must put a space between each attribute's value and the next attribute's name.
- XML does not support special characters such as ampersands (&). To include an ampersand within a tag, you must use the code `&amp` (for instance, to use ampersands in menu items or other UI elements).

Encoding characters

In previous releases of the Extension Manager, you had to specify the document encoding type for your documents. Beginning with version 1.6, the Extension Manager uses UTF-8 document encoding by default.

MXI tags

The following table lists the primary tags available in the MXI file, briefly describes each tag, and specifies whether the tag contains child tags. Use this table to get an overview of what tags are available and what functions they perform.

Tag	**Description**	**Contains Child Tags?**
macromedia-extension	Main tag for extension installation file.	Yes
defaultLanguage	Default for multilingual extensions (version 2.1 only).	No
author	Name of the extension's author.	No
description	Describes what the extension does.	No
products	Container tag that contains tags specifying an extension's compatibility.	Yes
license-agreement	Allows a third-party developer to include a license agreement that is displayed at installation.	No
ui-access	Specifies the text that will appear in the Extension Manager window when the extension is selected.	No
files	Container tag that contains tags describing the files an extension installs.	Yes
configuration-changes	Container tag for tags that modify the application's configuration. These include menus, shortcuts, server behaviors, and data sources.	Yes

Tag	Description	Contains Child Tags?
documenttype-changes	Describes changes made to the MMdocumentTypes.xml file.	Yes
toolpanel-changes	Marks the beginning of Flash toolpanel changes.	Yes
ftp-extension-map-changes	Specifies a change to the FTPExtensionMap.txt file. This defines whether the file is downloaded or uploaded as an ASCII or binary file from Dreamweaver to an FTP server.	Yes
insertbar-changes	Specifies changes to be made to the insertbar.xml file and add new toolbars files.	Yes
server-behavior-changes	Container tag for changes to menus in the menus.xml file in any of the Dreamweaver MX Configuration/ ServerBehaviors/document_type folders.	Yes
server-format-changes	Container tag for changes to menus in the menus.xml file in any of the Dreamweaver MX Configuration/ ServerFormats/document_type folders	Yes
data-source-changes	Container tag for changes to menus in the menus.xml file in any of the Dreamweaver MX Configuration/ DataSources/document_type folders.	Yes
menu-insert	Provides information about a menu bar, menu, menu item, or format to remove during installation of an extension.	No
menu-insert	Specifies where in the product's menus to insert a menu bar, menu, menu item, or format during installation of an extension.	No
menu	Describes a menu or submenu to be inserted into the product's menu structure during installation of an extension.	No
menuitem	Describes the menu item to be inserted into the product's menu structure during installation of an extension.	No
format	Describes the data format to be inserted into the Dreamweaver Format menu during installation of the extension.	No
separator	Specifies that a separator be inserted into a menu at the location indicated by the containing `menu-insert` tag.	No
comment	Provides a comment about an item being inserted into the menu structure. The Extension Manager inserts this comment (in the form of an XML comment tag) into the menus.xml file as it installs the extension.	No
shortcut-remove	Indicates that the specified keyboard shortcut should be removed from the menus.xml file.	No
shortcut-insert	Indicates that a keyboard shortcut should be added to the menus.xml file.	No

Tag	Description	Contains Child Tags?
shortcut	Specifies a keyboard shortcut to be added to the menus.xml file.	No
taglibrary-changes	Describes changes to be made to the TagLibraries.vtm file.	Yes
toolbar-changes	Inserts the specified tag library at the end of the file.	Yes
extensions-changes	Container tag that describes any changes to the Extensions.txt file, such as adding or removing extensions that you can open in Dreamweaver.	Yes
file-tokens	Container tag that allows you to specify tokens. Tokens let you specify the destination folder of one or more files from your extension during installation or provide a dialog box for the user to choose a destination folder for certain files.	Yes
token	Defines a custom token for an extension. Custom tokens let you specify the destination folder of one or more files from an extension during installation, or provide a dialog box for the user to choose a destination folder.	No

MXI tag descriptions

The tags used in the extension installation file are described below. Attribute names enclosed in curly braces (`{ }`) are optional. The tags are listed according to their position with the MXI file hierarchy. For example, the macromedia-extension tag is the main tag within the file, and is the first tag described.

macromedia-extension

Description

Main tag for extension installation file.

Attributes

`id`, `name`, `version`, `{type}`, `{requires-restart}`, `{ismultilingual}`, `{name_resid}`

`id` A unique extension ID, to be created by Adobe after you submit your extension. Never add or modify this attribute yourself.

`name` The name of the extension. This must be a `VARCHAR` data type with a limit of 255 characters.

`version` The version number of the extension, in the format *a*`{.`*b*`{.`*c*`}}`, where *a*, *b*, and *c* are all positive integers. For example, valid version numbers include 1, 3.6, and 10.0.1. The first number is the major version number, incremented when you make substantial changes to the extension; the second number is the minor version number, incremented for smaller changes. The third number is incremented for each new “build” of the extension between releases; for example, after you submit version 4.1 of your extension to Adobe, it may be returned to you for minor corrections. You might label the fixed version 4.1.1; after a couple of rounds of corrections, the version number of the posted extension might be 4.1.3.

`mxiversion` Attribute for version 2.x that degrades gracefully. Indicates the version of the MXI specification. Extension Manager 2.0 supports mxiversion 2.0 and earlier, while Extension Manager 2.1 supports mxiversion 2.1 and earlier. If mxiversion is unspecified, the default value is 1.0.

Note: If the specified mxiversion is more recent than the current Extension Manager, an alert appears during installation, indicating that a newer Extension Manager is required.

`xmanversion` Attribute for version 2.x that degrades gracefully. Indicates the oldest version of Extension Manager that can install this extension.

Note: Specify the `xmanversion` attribute only if a newer version of Extension Manager doesn’t support the installer extension file. To indicate the product versions that are compatible with an extension, see the `maxversion` attribute for “product” on page 10.

`icon` Attribute for version 2.x that does not degrade gracefully. Indicates the path to customized extension icon displayed by Extension Manager. To specify this relative path, use the `$ExtensionSpecificEMStore` attribute. For more information, see the `destination` attribute for “file” on page 13. If `icon` is unspecified, the default icon is used.

Note: The `icon` attribute applies only to CS4 products. To specify an icon for CS3 or earlier products, use the `type` attribute.

`type` Indicates the kind of extension. This optional attribute applies only to Dreamweaver, Fireworks, and Flash.

- Valid values for Dreamweaver are:

  `behavior`, `browserprofile`, `codehint codesnippet`, `coloringscheme`, `command`, `connection`, `datasource`, `dictionary`, `documenttype`, `encoding`, `flashbuttonstyle`, `flashelement`, `floater`, `insertbar`, `jsextension`, `keyboard shortcut`, `object`, `plugin`, `propertyinspector`, `report`, `referencebook`, `samplecontent`, `serverbehavior`, `serverformat`, `servermodel`, `site`, `suite`, `taglibrary`, `template`, `thirdpartytags`, `toolbar`, `translator`, `utility`, and `query`.

- Valid values for Fireworks are:

  `autoshape`, `command`, `commandpanel`, `dictionary`, `keyboard shortcut`, `library`, `pattern`, and `texture`.

- Valid values for Flash are:

  `actionscript`, `flashcomponent`, `flashcustomaction`, `flashimporter`, `flashpanel`, `flashtemplate`, `generatorobject`, `keyboardshortcut`, `lesson`, `library`, `publishtemplate`, `sample`, `smartclip`, and `utility`.

  Extensions of type `"generatorobject"` are supported only by Flash 5 and earlier. Values are not case-sensitive; `"object"` is equivalent to `"Object"`.

Note: The value "`suite`" denotes a set of items that are released as a unit, in a single MXP file, with a single MXI file. For example, you can create a set of objects, a command, a palette, and behaviors to make a process such as layer alignment easier to complete. Specify a single name and version for the entire suite.

`requires-restart` Indicates whether the Adobe application must be restarted after the extension is installed. Valid values are "`true`" and "`false`".

`ismultilingual` If this attribute is false, all multilingual elements are ignored. If it is true, multilingual elements install language-specific files and apply related configuration changes. If this attribute is not specified, it is considered false. For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

`name_resid` References string with value of "name_resid" in resource file (see `isresourcefile` attribute for "file" on page 13), and displays that string in the "extension" field of the user interface. For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

This tag must contain a `description` tag, a `ui-access` tag, a `products` tag, and an `author` tag. If you're changing the menus, this tag must also contain a `configuration-changes` tag. If you're installing files, this tag must also contain a `files` tag.

Container

None.

Example

```
<macromedia-extension
  name = "FrobSquigger Command"
  version = "1.0.0"
  type = "command"
  requires-restart = "false" >
  mxiversion = "2.0"
  xmanversion = "2.0"
  icon = "command.png"
  <!-- description, ui-access, products, author, configuration-changes, and
  files tags go here -->
</macromedia-extension>
```

Note: The `macromedia-extension` tag must be located at line 1 of your file.

defaultLanguage

Description

This tag specifies the default language for installed files. Extension Manager determines the correct language by completing these steps, listed in order of priority:

1 The language of the point product (defined in XManConfig.xml file for point product).

2 The language of the Extension Manager interface.

3 The language selected by the user when prompted by Extension Manager.

4 If the extension doesn't provide files for the user-selected language, Extension Manager installs files specified by the "defaultLanguage" tag.

If Extension Manager can't find any files belonging to the above languages, it installs files for all languages.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Possible values for the defaultLanguage tag are listed below:

Language	Value
American English	en_US
British English	en_GB
Danish	da_DK
Dutch	nl_NL
Finish	fi_FI
French	fr_FR
German	de_DE
Italian	it_IT
Norwegian	nb_NO
Portugese	pt_BR
Spanish	es_ES
Catalan	ca_ES
Swedish	sv_SE
Ukranian	uk_UK
Chinese	zh_CN
Taiwanese	zh_TW
Japanese	ja_JP
Korean	ko_KR

Example

```
<defaultLanguage>fr_FR</defaultLanguage>
```

description

Description

Describes what the extension does or is used for.

Attributes

{href}, {`resid`}, {source}

`href` Attribute for version 2.x that degrades gracefully. Indicates online URL that will be displayed as the description of the extension. The value must start with either "http://" or "https://".

`resid` References string with value of "resid" in resource file. When users select an extension in Extension Manager, this string is displayed in the lower-right area of the user interface. For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

`source` Attribute for version 2.x that does not degrade gracefully. Indicates the relative path to the HTML file on the local computer, specified by the `$ExtensionSpecificEMStore` attribute. For more information, see the `destination` attribute for "file" on page 13.

Note: If href is specified and computer is online, Extension Manger displays the URL in the description field. If source is specified, the local path is displayed for the description. If neither href nor source are specified, text in the Contents tag below is displayed.

Contents

This tag must contain a `CDATA` section, which you can format with any HTML tags. If text colors are unspecified, the background is gray (#626262), and the text is black.

Container

This tag must be contained in a `macromedia-extension` tag.

Example

```
<description><![CDATA[This command converts a frob into a squig.<br>
Be sure not to use it on a grickle!]]></description>
```

license-agreement

Description

This tag lets third-party developers include a license agreement with extensions they develop. The contents of this tag are displayed under the heading Third Party License, at the end of the new extension install license.

If the `license-agreement` tag is omitted from the MXI file, only the default extension disclaimer is displayed.

Attributes

`{resid}`

`resid` References string with value of "resid" in resource file and displays that string after the License Agreement when installing the extension. For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

This tag must contain a `CDATA` section, which you can format with any HTML tags. If text colors are unspecified, the background is gray (#585858) and the text is nearly white (#E0E0E0).

To display double-byte characters, the content of the CDATA section needs to include "charset=UTF-8." For example:

```
<description>
  <![CDATA[<meta http-equiv=Content-Type content="text/html;charset=UTF-8">
  <br>
  This is a sample Exchange item.<br>
  It is a sample library containing a single button.]]>
</description>
```

Container

This tag must be contained in the `macromedia-extension` tag.

Example

```
<license-agreement><![CDATA[You are about to install an Extension from the
  Adobe Exchange. The Adobe Exchange is an area of the adobe.com website that
  allows third parties to submit extensions for posting to adobe.com.]></
  license agreement>
```

ui-access

Description

Specifies the text that will appear in the Extension Manager window when the extension is selected. You should include information about where to find the item in the product's user interface as well as a brief description of the item's use.

If the `href` or `source` attributes are specified, Extension Manager displays the HTML content specified by those attributes. If those attributes are unspecified, ui-access is appended to the contents of the description tag.

Attributes

`{resid}`

`resid` References string resource with value of "resid" in resource file and displays that string in lower-right part of user interface. For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

This tag must contain a `CDATA` section. You can use `br` and `&nbsp;` to format the `CDATA` information. This is a `VARCHAR` data type with a limit of 512 characters.

Container

This tag must be contained in a `macromedia-extension` tag.

Example

```
<ui-access><![CDATA[You can run this command by choosing<br>
&nbsp;&nbsp;&nbsp;Commands > Convert Frob to Squig.]]></ui-access>
```

products

Description

Container tag for `product` tags.

Attributes

None.

Contents

This tag must contain one or more `product` tags.

Container

This tag must be contained in a `macromedia-extension` tag.

Example

```
<products>
```

```
    <!-- product tags go here -->
</products>
```

product

Description

Specifies which Adobe product or products your extension is compatible with. List each product in a separate `product` tag.

Attributes

`name`, `{version}`, `{primary}`, `{required}`, `{maxversion}`, `{familyname}`

`name` The name of an Adobe product. This attribute uses a `VARCHAR2` data type with a limit of 64 characters. Valid values appear below:

- `Bridge`
- `Contribute`
- `Dreamweaver`
- `Fireworks`
- `Flash`
- `Illustrator`
- `InCopy`
- `InDesign`
- `Photoshop32` (Windows 32-bit)
- `Photoshop64` (Windows 64-bit)

Note: To specify Photoshop for Mac OS and Windows, see the “familyname” attribute below. When that attribute is specified, the “name” attribute isn’t required.

`version` The version number of the specified Adobe product. Valid version numbers are:

Product	**Version number**
Bridge CS4	3
Contribute CS4	5
Dreamweaver MX 2004	7
Dreamweaver 8	8
Dreamweaver CS3	9
Dreamweaver CS4	10
Fireworks MX 2004	7
Fireworks 8	8
Fireworks CS3	9
Fireworks CS4	10
Flash MX 2004	7
Flash 8	8
Flash CS3	9

Product	Version number
Flash CS4	10
Illustrator CS4	14
InCopy CS4	6
InDesign CS4	6
Photoshop CS4	11

For example, if your extension is for Dreamweaver CS4, specify `version = "10"`. The extension can be installed in any version of the product greater than or equal to the specified version number. This attribute uses a `VARCHAR` data type with a limit of 8 characters.

`primary` Indicates whether the specified Adobe product is the one the extension was primarily intended to be used with. For example, if the extension's user interface appears in Dreamweaver but the extension also uses Fireworks, Dreamweaver is the primary product. For example, `<product name = "Dreamweaver" version = "7" primary = "true" />` indicates that this extension is primarily intended for Dreamweaver; however, it might be used in another product that supports the Extension Manager. (If you set the primary attribute to "true" for multiple products, Extension Manager can install the extension into each product.)

`required` Indicates whether the specified Adobe product is required for the extension to function properly. If the extension will function without the indicated product, even if it won't function as well without it, specify `"false"` or omit the `required` attribute. If you don't specify `required = "true"` for any `product` tag, the product specified in the first `product` tag listed is assumed to be required.

For example, `<product name = "Dreamweaver" version = "7" required = "true" />` indicates that Dreamweaver is necessary to use this extension.

`maxversion` Specifies the latest product version that an extension can be installed in. For example, if a Dreamweaver extension can be installed with CS3 but not CS4, you would specify `<product name="Dreamweaver" version="9" maxversion="9"/>`

`familyname` Combines Photoshop products together. For example, to combine the standard and Extended versions, specify `<product version="10" familyname="Photoshop" primary="true" />`

Contents

None.

Container

This tag must be contained in a `products` tag.

Example

```
<product name = "Dreamweaver" version = "7" primary = "true" />
```

author

Description

Name of the author of the extension.

Attributes

`name, {author_resid}`

`name` The author’s name. This attribute uses a `VARCHAR` data type with a limit of 255 characters.

`author_resid` References string resource with value of 'author_resid' in resource file and displays that string in “Author” field of the user interface. For more information, see “Creating multilingual extension installers (version 2.1 only)” on page 46.

Contents

None.

Container

This tag must be contained in a `macromedia-extension` tag.

Example

```
<author name = "Jambalaya Joe"/>
```

files

Description

Container tag for all of the `file` tags.

Attributes

`{xml:lang}`

`xml:lang` Specifies the language for the group of files. Extension Manager compares this language with the user language, which is determined by the process outlined in “defaultLanguage” on page 6. If the languages match, the files are installed; if not, the files are ignored. If Extension Manager can't determine the user language, it copies all files regardless of their specified language.

For more information, see “Creating multilingual extension installers (version 2.1 only)” on page 46.

Contents

This tag must contain one or more `file` tags.

Container

This tag must be contained in a `macromedia-extension` tag.

Example

```
<files>
  <!-- file tags go here -->
</files>
```

file

Description

Provides information about a specific file to be installed as part of the extension.

Note: Use `menu-insert` tags to explicitly add your item to menus even if your extension is an object or a command; don’t rely on the Adobe product to automatically add objects and commands to its menus. See `menu-insert` for details.

Attributes

`source`, `destination`, `{platform}`, `{shared}`, `{systemfile}`, `{win-extension}`, `{isresourcefile}`, `{minVersion}`, `{maxVersion}`

`source` The name of the file. It can include a path relative to the location of the installation file; the extension’s files don’t all have to be in the same folder. The filename must be a valid name in both Windows and Mac OS, unless you specify a value for the `platform` attribute. You can use a colon (:), slash (/), or backslash (\) as the separator between folder names (and before the filename) in the path. Note that in some operating systems, filenames are case-sensitive; make sure to use the same capitalization in the `source` attribute as you use for the corresponding file and folder names on your disk. Filenames should be a total of 30 characters or less.

Do not use the same filenames as Adobe extensions unless your extension is intended as a substitute for a Adobe extension.

To create an extension as part of a bundle or framework in Mac OS, use either of the following formats, without wildcards:

- `<files><file source="sourceFolder" destination="$photoshop/" /></files>`
- `<files><file source="sourceFolder/" destination="$photoshop/" /></files>`

`destination` The name of the folder the file will be copied to. If the folder doesn’t exist, the Extension Manager creates it during installation. Note that this attribute should contain a folder name, not a filename. The filename is specified by the `source` attribute.

Attributes you can use to refer to installation folders include the following:

Note: Photoshop CS4, Illustrator CS4, and InDesign CS4 define many additional attributes in the XManConfig.xml file, located in the Configuration folder in the application folder.

Various applications and system folders

Attribute	Description
$Dreamweaver	Specifies the Dreamweaver installation folder
$Fireworks	Specifies the Fireworks installation folder
$Flash	Specifies the Flash installation folder
$System	Specifies the System or System32 folder
$Fonts	Specifies the Font folder on the computer’s hard disk
$ExtensionSpecificEMStore	Attribute for version 2.x that cannot degrade gracefully. Specifies the folder that stores extension-specific file.

Photoshop

Attribute	Description
$photoshopappfolder	Specifies the installation folder
$pluginsfolder	Specifies the top level of the Plug-ins folder
$presetsfolder	Specifies the top level of the Presets folder
$scripts	Specifies the Scripts folder
$actions	Specifies the Actions folder
$brushes	Specifies the Brushes folder
$matlab	Specifies the MATLAB folder

Adobe Bridge

Attribute	Description
$bridgeappfolder	Specifies the Bridge installation folder
$pluginsfolder	Specifies the Bridge Plug-ins folder
$presetsfolder	Specifies the Bridge Presets folder
$bridge	Specifies the Bridge ExtensionManager Config folder
$startupscripts	Specifies the global (suite) startup scripts folder
$bridgestartupscripts	Specifies the Bridge global startup scripts folder
$extensions	Specifies the Bridge namespace extensions folder
$workspaces	Specifies the Bridge user workspaces folder
$extensionworkspaces	Specifies the Bridge namespace extensions workspaces folder
$userscripts	Specifies the Bridge user startup scripts folder

Illustrator

Attribute	Description
$illustrator	Specifies the installation folder
$plugin	Specifies the Plug-ins folder
$scripting	Specifies the Scripting folder
$presets	Specifies the Presets folder

The Extension Manager picks the appropriate system and font folder on the user's disk, based on the user's platform and operating system. If none of these options suits your needs, you can define your own custom tokens for the destination of your files. For information about writing custom tokens, see `token` on page 41.

Generally, destination folders should be inside the application's Configuration folder. The `destination` attribute is not case-sensitive; `configuration` is the same as `Configuration`. The folder name must be a valid name on both Windows and Mac OS, unless you specify a value for the `platform` attribute. You can use a colon (:), slash (/), or backslash (\) as the separator between folder names in the path. Note that in some operating systems, folder names are case-sensitive; make sure to use the correct capitalization for your folder names.

If your extension for Dreamweaver contains multiple files, such as help files, many images, or a suite of items, the destination folder for your files should be a folder in the Configuration/Shared folder. The folder name should be related to your company or product—for example, Configuration/Shared/MagicTricks.

You do not need to include the Configuration folder in the path name for Flash extensions. The folder specified in the `destination` tag is automatically created in the Configuration folder if it does not already exist.

`platform` Indicates what platform the file is intended for. If you specify a platform, the file is installed only on that platform; for instance, you can provide two versions of a file, one for Windows and one for Mac OS, and specify a platform value for each. Valid values are `"win"` and `"mac"`. If you don't specify this attribute, the file is installed on both platforms.

`shared` Indicates whether the file is used by more than one extension. When you use the Extension Manager to remove an extension, a shared file associated with that extension is not deleted as long as other installed extensions refer to that file. Valid values are `"true"` and `"false"`. If you don't specify this attribute, its default value is `"false"`.

Note: If you install a newer version of a shared file and another extension is using the old version of the file, the new shared file either must be backward compatible with the other extension, or must have a new filename so that the other extension continues to work properly.

`systemfile` Indicates whether the file is used by anything other than extensions. For example, some extensions provide new versions of DLLs or other system files, or files that are used by other applications. If a file is specified as `systemfile = "true"`, it is not deleted when the extension is removed, even if no other extensions use the file. When `systemfile` is set to true, the `shared` attribute is ignored.

`win-extension` Used when a file is generated on Mac OS that does not include the Windows extension, such as .fla or .htm.

For example, a FLY file named "shoo" created on Mac OS installs with the correct creator and file-type information on another Mac OS. However, in a Windows system, it requires the following to install properly in the Configuration\Shared\Thingies folder:

```
<file source="shoo" destination="$Dreamweaver/configuration/shared/thingies/"
win-extension="fly" />.
```

This adds the extension to the filename and installs "shoo.fly".

If you create a file in Windows that does include the extension, such as "heeble.fla" or "frob.htm", and install it on Mac OS, the `win-extension` attribute does not need to be added to the file tag.

Note: If the `platform` attribute is included, the `win-extension` attribute is ignored.

`isresourcefile` If set to `true`, this attribute flags the file as a resource file containing language-specific text strings. Place resource files in a folder with the name [*installer prefix*].mxi_Resources. When the MXI file is loaded, Extension Manager copies this folder into following location, where it then looks for text strings:

- Win XP: C:\Documents and Settings\All Users\Application Data\Adobe\Extension Manager2
- Mac OS: /Library/Application Support/Adobe/Extension Manager2

If this attribute is not specified, it is considered false.

For more information, see “Creating multilingual extension installers (version 2.1 only)” on page 46.

`minVersion, maxVersion` Specifies the minimum and maximum version of the product to which the file will be installed. For example, if minVersion is 9 and maxVersion is 10, the file won't be installed in product version 8 or 11, but will be in product version 9. The correct format for this attribute is Major_Version_Number.Minor_Version_Number.Build_Number.

Contents

None.

Container

This tag must be contained in a `files` tag.

Example

```
<file source = "frob2squig.htm" destination = "$Dreamweaver/configuration/
  commands/common" platform = "mac" shared = "false" />
```

configuration-changes

Description

Container tag for changes to menus, shortcuts, server behaviors, server formats, and data sources.

Attributes

None.

Contents

This tag may contain any combination of `data-source-changes`, `menu-insert`, `menu-remove`, `server-behavior-changes`, `server-format-changes`, `server-format-definition-changes`, `shortcut-insert`, and `shortcut-remove` tags.

Container

This tag must be contained in a `macromedia-extension` tag.

Example

```
<configuration-changes>
  <!-- ftp-extension-map-changes, menu-remove, menu-insert, shortcut-remove,
  shortcut-insert, server-behavior-changes, server-format-changes, server-
  format-definition-changes, and data-source-changes tags go here -->
</configuration-changes>
```

documenttype-changes

Description

Describes changes to be made to the MMDocumentTypes.xml file.

Attributes

None.

Contents

This tag contains the `documenttype-insert` and `documenttype-remove` tags.

Container

This tag must be contained in a `configuration-changes` tag.

Example

This example illustrates the syntax of the tags that can be contained by the `documenttype-changes` tag.

```
<documenttype-changes>
  <documenttype-insert>
    <documenttype>
      ...
    </documenttype>
  </documenttype-insert>
  <documenttype-remove id="remove_id" />
</documenttype-changes>
```

documenttype-insert

Description

Insert the specified tag library at the end of file.

Attributes

`{xml:lang}`

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

The `documenttype` tag describes the document type to be inserted. The Extension Manager verifies only that the XML structure is valid.

Container

This tag must be contained in a `documenttype-changes` tag.

Example

```
<documenttype-insert>
  <documenttype>
    ...
  </documenttype>
</documenttype-insert>
```

documenttype-remove

Description

Removes the specified document type.

Attributes

`id, {xml:lang}`

`id` ID of the document type to remove.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

None.

Container

This tag must be contained in a `documenttype-changes` tag.

Example

```
<documenttype-remove id="remove_id" />
```

toolpanel-changes

Description

Marks the beginning of Flash tool panel changes.

Attributes

None.

Contents

This tag may contain the `toolpanel-item-insert` tag.

Container

This tag must be contained in a `configuration-changes` tag.

Example

```
<configuration-changes>
      <toolpanel-changes>
            ...
      </toolpanel-changes>
</configuration-changes>
```

toolpanel-item-insert

Description

Inserts the tool with the specified name into the Flash tool panel.

Attributes

`name, position, depth, {xml:lang}`

`name` The name of the tool to insert. This is a required attribute.

`position` The 0-based position at which to insert the tool. Valid positions are 0 through 17. If the attribute is missing, or has a value beyond 17, the tool assumes the last position in the toolbar by default. This is an optional attribute.

`depth` The 0-based depth of the tool is its position in the toolbar, where 0 specifies the top. If the depth attribute is missing, or has a value beyond the bottom of the menu as the tool's position, the tool depth defaults to the bottom of the menu. This is an optional attribute.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

None.

Container

This tag must be contained in a `toolpanel-changes` tag.

Example

```
<toolpanel-changes>
      <toolpanel-item-insert name="polystar" position="7" />
</toolpanel-changes>
```

ftp-extension-map-changes

Description

Specifies a change to the FTPExtensionMap.txt file located in the Configuration folder.

Attributes

None.

Contents

This tag may contain an `ftp-extension-insert` tag and an `ftp-extension-remove` tag.

Container

This tag must be contained in a `configuration-changes` tag.

Example

```
<ftp-extension-map-changes>
  <!-- ftp-extension-insert, ftp-extension-remove tags go here-->
</ftp-extension-map-changes>
```

ftp-extension-insert

Description

Specifies a change to the FTPExtensionMap.txt file. This defines whether the file is downloaded or uploaded as an ASCII or binary file from Dreamweaver to an FTP server.

Attributes

`extension`, `type`, `mac-creator`, `mac-file-type`

`extension` The file extension, such as .gif or .jpg.

`type` The format used when you upload a file to the FTP server. The current valid values are `"ASCII"` and `"Binary"`.

`mac-creator` The Mac OS creator code. If you do not know the creator code, use “`????`”.

`mac-file-type` The Mac OS file type. If you do not know the file type, use “`????`”.

Contents

None.

Container

This tag must be contained in an `ftp-extension-map-changes` tag.

Example

```
<ftp-extension-insert extension="JPG" mac-creator ="MKBY" mac-file-type="JPEG"
  type="ASCII"/>
```

ftp-extension-remove

Description

Indicates the extension that is removed from SourceFormat.txt in the Configuration folder.

Attributes

`extension`

`extension` The file extension, such as .gif or .jpg.

Contents

None.

Container

This tag must be contained in an `ftp-extension-map-changes` tag.

Example

```
<ftp-extension-remove extension="JPG" />
```

insertbar-changes

Description

Marks the beginning of changes to Insertbar.xml. Note that InsertBar.xml is automatically updated when objects are installed into Dreamweaver MX. Modifying the file explicitly from the MXI file is not required.

Attributes

None.

Contents

The `insertbar-insert`, `insertbar-item-insert`, and `category` tags describe the category to be inserted. The Extension Manager verifies only that the XML structure is valid.

Container

This tag must be contained in a `configuration-changes` tag.

Example

```
<insertbar-changes>
  <insertbar-insert insertBefore|insertAfter="category_id">
    <category ...>
      <itemtype.../>
    </category>
  </insertbar-insert>
  <insertbar-remove id="category_id" />
  <insertbar-item-insert
    insertBefore|insertAfter|appendTo|prependTo="category_or_item_id"
    category="category_id"> <itemtype.../>
  </insertbar-item-insert>
  <insertbaritem-remove id="item_id" />
</insertbar-changes>
```

insertbar-insert

Description

Inserts the specified category at the end of file.

Attributes

`insertBefore | insertAfter, {xml:lang}`

`insertBefore | insertAfter = category_id` of the existing category to insert before or after. You can specify only one of the two attributes; either `insertBefore` or `insertAfter`.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

The `category` tag that describes the category to insert. The Extension Manager verifies only that the XML structure is valid.

Container

This tag must be contained in a `insertbar-changes` tag.

Example

```
<insertbar-insert insertBefore | insertAfter = category_id>
  <category ...>
    <itemtype.../>
  </category>
</insertbar-insert>
```

insertbar-remove

Description

Removes the specified category.

Attributes

`category_id, {xml:lang}`

`category_id` ID of the category to be removed.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

None.

Container

This tag must be contained in the `insertbar-changes` tag.

Example

```
<insertbar-remove id="category_id" />
```

insertbar-item-insert

Description

Inserts the specified item at the specified location.

Attributes

`insertBefore | insertAfter`, `appendTo | prependTo`, `category`, `{xml:lang}`

`insertBefore | insertAfter` ID of the existing item before or after which the specified item should be inserted.

`appendTo | prependTo` ID of the existing category to which the specified item is appended or prepended.

`category` ID of the category to append to if the `insertBefore|insertAfter` item isn't found.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

A tag that describes the item to insert. The Extension Manager verifies only that the XML structure is valid.

Container

This tag must be contained in the `insertbar-changes` tag.

Example

```
<insertbar-item-insert
  insertBefore | insertAfter | appendTo | prependTo = category_or_item_id
  category = category_id
  <itemtype.../>
</insertbar-item-insert>
```

insertbar-item-remove

Description

Removes the specified `insertbar` item.

Attributes

`id, {xml:lang}`

`id` ID of the item to be removed.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

None.

Container

This tag must be contained in a `insertbar-changes` tag.

Example

```
<insertbar-item-remove id = item_id />
```

server-behavior-changes

Description

Container tag for changes to menus in the ServerBehaviors.xml file in any of the Dreamweaver MX Configuration/ServerBehaviors/*servermodel* folders.

Attributes

`servermodelfolder`

`servermodelfolder` The name of the server model folder in which the changes are to be made. The name of any installed server model (such as `"ASP.NET_Csharp"`, `ASP.NET_VB"`, `"ASP_Js"`, `"ASP_Vbs"`, `"ColdFusion"`, `"UD4-ColdFusion"`, `"PHP_MySQL"` or `"JSP"`) is a valid value. Note that the attribute value must precisely match the name of the corresponding server model folder.

Contents

This tag may contain any combination of `menu-remove` and `menu-insert` tags.

Container

This tag must be contained in a `configuration-changes` tag.

Example

```
<server-behavior-changes servermodelfolder = "ASP_VB">
  <!-- menu-remove, menu-insert tags go here -->
</server-behavior-changes>
```

server-format-changes

Description

Container tag for changes to menus in the Formats.xml file in any of the Dreamweaver MX Configuration/ServerFormats/*servermodel* folders.

Attributes

`servermodelfolder`

`servermodelfolder` The name of the server model folder in which the changes are to be made. The name of any installed server model (such as `"ASP.NET_Csharp"`, `ASP.NET_VB"`, `"ASP_Js"`, `"ASP_Vbs"`, `"ColdFusion"`, `"UD4-ColdFusion"`, `"PHP_MySQL"` or `"JSP"`) is a valid value. Note that the attribute value must precisely match the name of the corresponding server model folder.

Contents

This tag may contain any combination of `menu-remove` and `menu-insert` tags.

Container

This tag must be contained in a `configuration-changes` tag.

Example

```
<server-format-changes servermodelfolder = "ASP_VB">
  <!-- menu-remove, menu-insert tags go here -->
</server-format-changes>
```

server-format-definition-changes

Description

Container tag for changes to menus in the ServerFormats.xml file in any of the Dreamweaver MX Configuration/ServerFormats/*servermodel* folders.

Attributes

`servermodelfolder`

`servermodelfolder` The name of the server model folder in which the changes are to be made. The name of any installed server model (such as `"ASP.NET_Csharp"`, `ASP.NET_VB"`, `"ASP_Js"`, `"ASP_Vbs"`, `"ColdFusion"`, `"UD4-ColdFusion"`, `"PHP_MySQL"` or `"JSP"`) is a valid value. Note that the attribute value must precisely match the name of the corresponding server model folder.

Contents

This tag may contain any combination of `menu-remove` and `menu-insert` tags.

Container

This tag must be contained in a `configuration-changes` tag.

Example

```
<server-format-definition-changes servermodelfolder = "ColdFusion">
  <!-- menu-remove, menu-insert tags go here -->
</server-format-definition-changes>
```

data-source-changes

Description

Container tag for changes to menus in the DataSources.xml file in any of the Dreamweaver MX Configuration/DataSources/*servermodel* folders.

Attributes

`servermodel`

`servermodelfolder` The name of the server model folder in which the changes are to be made. The name of any installed server model (such as `"ASP.NET_Csharp"`, `ASP.NET_VB"`, `"ASP_Js"`, `"ASP_Vbs"`, `"ColdFusion"`, `"UD4-ColdFusion"`, `"PHP_MySQL"` or `"JSP"`) is a valid value. Note that the attribute value must precisely match the name of the corresponding server model folder.

Contents

This tag may contain any combination of `menu-remove` and `menu-insert` tags.

Container

This tag must be contained in a `configuration-changes` tag.

Example

```
<data-source-changes servermodel = "ASP_VB">
  <!-- menu-remove, menu-insert tags go here -->
</data-source-changes>
```

menu-insert

Description

Specifies where in the product's menus to insert a menu bar, menu, menu item, or format during installation of this extension.

Use `menu-insert` tags to explicitly add your extension to menus even if your extension is an object or a command; don't rely on the Adobe product to automatically add objects and commands to its menus. To ensure that your extension is not automatically added to the menus, add `<!-- MENU-LOCATION=NONE -->` to the top of each of your extension's HTML files. If you do this, you must make an entry for your file in the menus.xml file.

Attributes

`insertAfter`, `insertBefore`, `appendTo`, `prependTo`, `{skipSeparator}`, `{xml:lang}`

You can specify only one of the following four attributes: `insertAfter`, `insertBefore`, `appendTo`, or `prependTo`.

`insertAfter` Indicates that the new item should be inserted immediately following the item with the specified ID. (The ID can be the ID of a menu bar, a menu, a menu item, or a format.)

Note: No menu can appear to the right of the Help menu in Dreamweaver. If you insert a new menu after the Help menu, the application displays the new menu to the left of the Help menu.

`insertBefore` Indicates that the new item should be inserted immediately before the item with the specified ID. (The ID can be the ID of a menu bar, a menu, a menu item, or a format.)

`appendTo` Indicates that the new item should be inserted immediately after the last item in the specified menu or menu bar. (The specified ID can be the ID of a menu bar or a menu only, not a menu item or format.)

`prependTo` Indicates that the new item should be inserted before the first item in the specified menu or menu bar. (The specified ID can be the ID of a menu bar or a menu only, not a menu item or format.)

`skipSeparator` Applies only if the `insertAfter` attribute is also specified. It indicates that the new item should be inserted after the separator that immediately follows the item specified in `insertAfter`. If there is no separator there, or if `insertAfter` is not used, this attribute is ignored. Valid values are `"true"` and `"false"`; the default value is `"false"`.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

In most uses, this tag must contain one or more `menu` tags and/or one or more `menuitem` tags. (When this tag appears inside a `server-format-definition-changes` tag, it contains `format` tags instead of `menuitem` tags.) It may also optionally contain `separator` tags and `comment` tags. Everything inside the `menu-insert` tag is inserted as a block, retaining its order; for example, if you list four menu items inside a `menu-insert` tag, those four items are inserted at the specified location so that they appear in the menu structure in the same order.

You can insert as many menus, menu items (or formats), separators, and comments as you want in a single `menu-insert` tag, but they can't be nested. That is, you can't insert a new menu and its contents by listing the items inside the `menu` tag. Instead, insert the menu first with one `menu-insert` tag, then insert all of the items into the new menu using another `menu-insert` tag. In the example below, the menu item Animals is inserted in the Insert menu after the Get More Objects menu item. The Animals menu item contains the submenus Cat and Dog. The Dog menu contains yet another submenu called Poodle. This is the resulting menu structure:

```
Insert
  ...
  Get More Objects
  Animals
    Dog
      Poodle
    Cat

  <menu-insert insertAfter="DWMenu_Insert_GetMoreObjects">
    <menu name="Animals" id="DWMenu_Insert_Animals" />
  </menu-insert>
  <menu-insert appendTo="DWMenu_Insert_Animals">
    <menu name="Dog" id="DWMenu_Insert_Animals_Dog" />
    <menuitem name="Cat" id="DWMenu_Insert_Animals_Cat" />
  </menu-insert>
  <menu-insert appendTo="DWMenu_Insert_Animals_Dog">
    <menuitem name="Poodle" id="DWMenu_Insert_Animals_Dog_Poodle" />
  </menu-insert>
```

Container

This tag must be contained in a `configuration-changes`, `server-behavior-changes`, `server-format-changes`, `server-format-definition-changes`, or `data-source-changes` tag.

Example

```
<menu-insert insertAfter = "DWMenu_Commands_SortTable" skipSeparator = "true">
  <!-- menu, menuitem (or format), separator, and comment tags here -->
</menu-insert>
```

menu-remove

Description

Provides information about a menu bar, menu, menu item, or format to remove during installation of this extension.

Note: If the user removes an installed extension, the menus, menu items, and formats that were removed when that extension was installed are not restored.

Attributes

`id, {xml:lang}`

`id` The menu ID of the item to be removed. Menu bars and menus are not removed unless they're empty. To find the menu ID of an item in a Dreamweaver menu, look for the item in the menus.xml files. (To find the menu ID of a format, look for the item in the Formats.xml files.)

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

None.

Container

This tag must be contained in a `configuration-changes`, `server-behavior-changes`, `server-format-changes`, `server-format-definition-changes`, or `data-source-changes` tag.

Example

```
<menu-remove id = "DWMenu_Commands_FrobSquigger-beta" />
```

menubar

Description

Provides information about a menu bar to be inserted into the product's menu structure during installation of this extension.

Attributes

`name`, `id`, `{platform}`

`name` The name of the menu bar to insert.

`id` ID for the new menu bar. Your menu IDs should start with a company name or some other namespace prefix to ensure uniqueness. In particular, don't start your menu IDs with `DW`, which is the prefix used by Dreamweaver menu IDs.

`platform` Indicates that the menu bar should appear only on the given platform. Valid values are `"win"` and `"mac"`.

Contents

None.

Container

This tag must be contained in a `menu-insert` tag.

Example

```
<menubar name = "Mugwump Context menu" id = “JMMugwumpContext"
platform = "mac"></menubar>
```

menu

Description

Describes a menu or submenu to be inserted into the product's menu structure during installation of an extension.

Attributes

`name`, `id`, `{platform}`

`name` The name of the menu to insert, as it will appear in the menu bar. To set the menu's access key (mnemonic) in Windows, use an underscore (_) in front of the access letter. The underscore is automatically removed on Mac OS.

`id` The menu ID of the new menu. Your menu IDs should start with a company name or some other namespace prefix to ensure uniqueness. In particular, don't start your menu IDs with `DW`, which is the prefix used by the Dreamweaver menu IDs. One useful approach is to prefix every ID with your domain name (with the elements reversed); for example, if your domain name is joe.com, you could start every ID with `com.joe.` to ensure uniqueness.

`platform` Indicates that the menu should appear only on the given platform. Valid values are `"win"` and `"mac"`.

Contents

None.

Note: Always use a `</menu>` tag to close a `<menu>` tag. Although the `<menu>` tag in the MXI file has no contents, it corresponds to the `<menu>` tag in the menus.xml file, which does have contents. Therefore, the `<menu>` tag is not defined as an empty tag, so you can't use the `/>` XML syntax to close the tag.

Container

This tag must be contained in a `menu-insert` tag.

Example

```
<menu name = "Recent Frobs Converted" id = "JMMenu_Commands_RecentFrobs"
platform = "mac" >
</menu>
```

menuitem

Description

Describes the menu item to be inserted into the product's menu structure during installation of this extension.

Attributes

`name`, `id`, `{key}`, `{platform}`, `{file}`, `{command}`, `{enabled}`, `{checked}`, `{dynamic}`, `{arguments}`, `{resid:name}`

`name` The menu item name that you want to appear in the menu. To set the menu item's access key in Windows, use an underscore (_) in front of the access letter. The underscore is automatically removed on Mac OS. If two menu items have the same access key, the access key works only for the first of the two.

Note: To make an underscore character appear in a menu item, precede it with a percent sign—that is, use `%_` instead of just an underscore.

`id` The menu ID of the new item. Your menu IDs should start with a company name or some other namespace prefix to ensure uniqueness. In particular, don't start your menu IDs with `DW`, which is the prefix used by the Dreamweaver menu IDs. One useful approach is to prefix every ID with your domain name (with the elements reversed); for example, if your domain name is joe.com, you could start every ID with `com.joe.` to ensure uniqueness.

`key` The shortcut key for the menu item. For syntax details that apply to Dreamweaver, see "About customizing Dreamweaver menus" in the "Customizing Dreamweaver" chapter of *Using Dreamweaver*.

`platform` Indicates that the menu should appear only on the given platform. Valid values are `"win"` and `"mac"`.

`file` The name of an HTML or JavaScript file that contains JavaScript code determining the behavior of the menu item. The path specified in the `file` attribute is relative to the Configuration folder. The `file` attribute overrides the `command`, `enabled`, and `checked` attributes. Either `file` or `command` must be specified for each menu item. Note that in some operating systems, filenames are case-sensitive; make sure to use the same capitalization in the `file` attribute as you use for the corresponding file and folder names on your hard disk.

`command` JavaScript code specifying the action to be taken when the user chooses the menu item.

`enabled` JavaScript code that the product executes before displaying the menu, to determine whether the menu item is enabled. The code should return a value of `true` or `false`, indicating that the menu item should be enabled or dimmed, respectively.

`checked` JavaScript code that the product executes before displaying the menu, to determine whether the menu item should have a check mark next to it. The code should return a value of `true` or `false`, indicating that the menu item should be checked or unchecked, respectively.

`dynamic` Indicates whether the menu item's text and state are to be determined dynamically, by an HTML file that contains JavaScript code (specified in the `file` attribute). Valid values are `"true"` and `"false"`. If you don't specify the `dynamic` attribute, its default value is `"false"`.

`arguments` Provides arguments to pass to the specified command file. This attribute is used only in conjunction with the `file` attribute.

`resid:name` References string with value of `resid:name` in resource file, and changes the Dreamweaver configuration file to display that string as a menu item. If Extension Manager can't find the string in the resource file, the value specified in the `name` attribute is displayed as the menu item.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

None.

Container

This tag must be contained in a `menu-insert` tag that is not inside a `server-format-definition-changes` tag.

Example

```
<menuitem name = "Convert Frobs to Squigs" id = "JMMenu_Commands_ConvertFrobs"
key = "Cmd+Alt+Shift+F" platform = "mac" file = "commands/common/
  frob2squig.htm"
dynamic = "false" />
```

format

Description

Describes the data format to be inserted into the Dreamweaver Format menu during installation of this extension.

Attributes

This tag's attributes are difficult to write by hand. The best way to create a `format` tag is to use the interface inside Dreamweaver. After you create a format, open the appropriate Formats.xml file in a text editor and copy the appropriate `format` tag (generated by Dreamweaver). Paste this tag into the appropriate place in your extension installation file.

Then add an `id` attribute. Each format ID must be unique; your IDs should start with a company name or some other namespace prefix to ensure uniqueness. In particular, don't start your IDs with `DW`, which is the prefix used by the Dreamweaver IDs. One useful approach is to prefix every ID with your domain name (with the elements reversed); for example, if your domain name is joe.com, you could start every ID with `com.joe.` to ensure uniqueness.

Contents

None.

Container

This tag must be contained in a `menu-insert` tag that is inside a `server-format-definition-changes` tag.

Example

```
<format
file = "Date/Time"
title = "Date/Time - 14:35"
expression = "<%\s*=\sDoDateTime\(.*, 4, 1033\)\s*%>"
strNamedFormat = "shortTime"
nLCID = 1033
id = "JMMenu_ServerFormatDef_ASP_2_DT18" />
```

separator

Description

Indicates that a separator should be inserted into a menu at the location specified by the containing `menu-insert` tag.

Attributes

`id, {platform}`

`id` The ID for the separator; each separator ID must be unique. Your separator IDs should start with a company name or some other namespace prefix to ensure uniqueness. In particular, don't start your IDs with `DW`, which is the prefix used by the Dreamweaver menu IDs. One useful approach is to prefix every ID with your domain name (with the elements reversed); for example, if your domain name is joe.com, you could start every ID with `com.joe.` to ensure uniqueness.

`platform` Indicates that the separator should appear only on the given platform. Valid values are `"win"` and `"mac"`.

Contents

None.

Container

This tag must be contained in a `menu-insert` tag.

Example

```
<separator id = "JMMenu_Commands_ConvertFrobs_Separator" platform = "win" />
```

comment

Description

Provides a comment about an item being inserted into the menu structure. The Extension Manager inserts this comment (in the form of an XML comment tag) into the menus.xml file as it installs the extension.

Attributes

None.

Contents

The text of a comment.

Container

This tag must be contained in a `menu-insert` tag.

Example

```
<comment>This command is part of the Mugwump extension.</comment>
```

shortcut-insert

Description

Indicates that a keyboard shortcut or shortcut list should be added to the menus.xml file.

Attributes

`list_Id, {xml:lang}`

`list_Id` The ID of the shortcut list into which the shortcut should be inserted. Use this attribute only when inserting a single shortcut into a list. Don't use it when inserting an entire list.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

This tag must contain a `shortcut` tag or a `shortcutlist` tag.

Container

This tag must be contained in a `configuration-changes` tag.

Example

```
<shortcut-insert list_Id = "DWMainWindow">
  <!-- shortcutlist or shortcut tag goes here -->
</shortcut-insert>
```

shortcut-remove

Description

Indicates that the specified keyboard shortcut be removed from the menus.xml file.

Attributes

`id, {xml:lang}`

`id` ID of the shortcut or shortcut list to remove. A shortcut list is removed only if it's empty.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

None.

Container

This tag must be contained in a `configuration-changes` tag.

Example

```
<shortcut-remove id = "DWMainWindow" />
```

shortcutlist

Description

Specifies a shortcut list to be added to the menus.xml file.

Attributes

`id`, `{platform}`

`id` ID for the new shortcut list. It should be the same as the menu ID for the menu bar that represents a window in Dreamweaver with which the shortcuts are associated. Currently supported IDs are `DWMainWindow`, `DWMainSite`, `DWTimelineInspector`, and `DWHTMLInspector`.

`platform` Indicates that the shortcut list should appear only on the given platform. Valid values are `"win"` and `"mac"`.

Contents

None.

Container

This tag must be contained in a `shortcut-insert` tag.

Example

```
<shortcutlist id = "NewMenuBar" platform = "win" />
```

shortcut

Description

Specifies a keyboard shortcut to be added to the menus.xml file.

`key`, `id`, `{command}`, `{file}`, `{platform}`

`key` The key combination used to activate the keyboard shortcut. For syntax details that apply to Dreamweaver, see “About customizing Dreamweaver menus” in the “Customizing Dreamweaver” chapter of *Using Dreamweaver*.

`id` A unique identifier for a shortcut. Your shortcut IDs should start with a company name or some other namespace prefix to ensure uniqueness. One useful approach is to prefix every ID with your domain name (with the elements reversed); for example, if your domain name is joe.com, you could start every ID with `com.joe.` to ensure uniqueness.

`command` The JavaScript code to execute when the user issues the keyboard shortcut.

`file` A file containing the JavaScript code to execute when the user issues the keyboard shortcut. The `file` attribute overrides the `command` attribute; either `file` or `command` must be specified for each shortcut.

`platform` Specifies that the shortcut works only on the indicated platform. Valid values are `"win"` and `"mac"`.

Contents

None.

Container

This tag must be contained in a `shortcut-insert` tag.

Example

```
<shortcut key = "Shift+F5" command = "dw.newDocument()" id = "ShortCutTest"
platform = "win" />
```

taglibrary-changes

Description:

Describes changes to be made to the `TagLibraries.vtm` file.

Attributes

None.

Contents

This tag may contain `taglibrary-insert` and `taglibrary-remove` tags.

Container

This tag must be contained in a `configuration-changes` tag.

taglibrary-insert

Description

Inserts the specified tag library at the end of file. Order is not important.

Attributes

`{xml:lang}`

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

This tag may contain `taglibrary` tags that describe a tag library to be inserted. The Extension Manager verifies only that the XML structure is valid.

Container

This tag must be contained in a `taglibrary-changes` tag.

taglibrary-remove

Description

Removes the specified tag library.

Attributes

`id, {xml:lang}`

`id` ID of the tag library to be removed.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents:

None.

Container

This tag must be contained in a `taglibrary-changes` tag.

toolbar-changes

Description

Marks the beginning of toolbar changes.

`{file}`

`file` This optional attribute specifies the name of the toolbar file to edit. If not supplied, the default is Toolbars.xml.

Contents

This tag may contain the `toolbar-insert`, `toolbar-item-insert`, `toolbar-remove`, and `toolbar-item-remove` tags.

Container

This tag must be contained in the `configuration-changes` tag.

Example

This example illustrates the syntax and hierarchy of all tags that can be contained by the `toolbar-changes` tag.

```
<toolbar-changes [file="file_name"]>
  <toolbar-insert>
    <toolbar ...>
    ...
    </toolbar>
  </toolbar-insert>
  <toolbar-remove id="toolbar_id" />
  <toolbar-item-insert
    insertBefore|insertAfter|appendTo|prependTo="toolbar_or_item_id"
    toolbar="toolbar_id">
    <itemtype.../>
  </toolbar-item-insert>
  <toolbar-item-remove id="toolbar_item_id" />
</toolbar-changes>
```

toolbar-insert

Description

Inserts the specified toolbar at the end of file.

Attributes

`{xml:lang}`

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

The `toolbar` tag, which describes the toolbar to be inserted. The Extension Manager verifies only that the XML structure is valid.

Container

This tag must be contained in the `toolbar-changes` tag.

Example

```
<toolbar-insert>
  <toolbar ...>
    ...
  </toolbar>
```

toolbar-remove

Description

Removes the specified toolbar.

Attributes

`id, {xml:lang}`

`id` ID of the toolbar to be removed.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

None.

Container

This tag must be contained in the `toolbar-changes` tag.

Example

```
<toolbar-remove id="toolbar_id" />
```

toolbar-item-insert

Description

Inserts the specified toolbar item at the specified location.

Attributes

`insertBefore|insertAfter`, `appendTo|prependTo`, `toolbar`, `{xml:lang}`

`insertBefore|insertAfter` ID of the existing toolbar item before or after which the specified item is inserted.

`appendTo|prependTo` ID of the existing toolbar to which the specified item is appended or prepended.

`toolbar` ID of the toolbar to append to if the `insertBefore|insertAfter` item isn't found.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

A tag that describes a toolbar item to be inserted. The Extension Manager verifies only that the XML structure is valid.

Container

This tag must be contained in the `toolbar-changes` tag.

Example

```
<toolbar-item-insert
  insertBefore|insertAfter|appendTo|prependTo="toolbar_or_item_id"
  toolbar="toolbar_id">
  <itemtype.../>
</toolbar-item-insert>
```

toolbar-item-remove

Description

Removes the specified toolbar item.

Attributes

`id`, `{xml:lang}`

`id` ID of the toolbar item to remove.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

None.

Container

This tag must be contained in the `toolbar-changes` tag.

Example

```
<toolbar-item-remove id="toolbar_item_id" />
```

extensions-changes

Description

Container tag that describes any changes to the Extensions.txt file, such as adding or removing extensions that you can open in Dreamweaver.

Attributes

None.

Contents

This tag may contain an `extension-insert` tag and an `extension-remove` tag.

Container

This tag must be contained in a `configuration-changes` tag.

Example

```
<extensions-changes>
  <!--extension-insert and extension-remove tags go here-->
</extensions-changes>
```

extension-insert

Description

Describes a new extension that Dreamweaver can open.

Attributes

`extension, description, {xml:lang}`

`extension` Specifies name of the extension, such as .gif or .htm.

`description` Describes what the extension is used for.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

None.

Container

This tag must be contained in an `extensions-changes` tag.

Example

```
<extension-insert extension="PHP" description="PHP files"/>
```

extension-remove

Description

Indicates an extension to remove from the Extensions.txt file.

Attributes

`extension, {description}, {xml:lang}`

`extension` The name of the extension, such as .gif or .htm.

`description` This optional tag allows you to specify a description of the extension being removed. If no description is provided, the extension is removed from all lines of the Extensions.txt file.

`xml:lang` Specifies the language for the listed file. Extension Manager compares this language with the user language, which is determined by the process outlined in "defaultLanguage" on page 6. If the languages match, configuration changes are applied; if not, they are ignored. If Extension Manager can't determine the user language, it applies all configuration changes regardless of their specified language.

For more information, see "Creating multilingual extension installers (version 2.1 only)" on page 46.

Contents

None.

Container

This tag must be contained in an `extensions-changes` tag.

Example

```
<extension-remove extension="PHP" description="PHP files"/>
```

or

```
<extension-remove extension="PHP" />
```

file-tokens

Description

Container tag that indicates any custom tokens.

Attributes

None.

Contents

One or more token tags for defining custom tokens.

Container

This tag must be contained in a `macromedia-extension` tag.

Example

```
 <file-tokens>
   <!-- token tags go here -->
</file-tokens>
```

token

Description

Defines a custom token for an extension.

Custom tokens let you specify the destination folder of one or more files from your extension during installation or provide a dialog box for the user to choose a destination folder for certain files. For example, you might use a custom token if your extension contains items that must be installed in a specific directory as well as a file, such as a tutorial, that can be installed anywhere on the hard disk. In this case, you could use a custom token tag to allow the user to select the destination folder for the tutorial while still installing the other files in the proper directories. If several files need to be grouped in the same directory, but that directory location is not important, you can allow the user to select the directory location.

Custom tokens are useful even if you don't allow the user to specify the destinations of files. You can easily change the destination directory of multiple files without having to manually change each destination path in the MXI file. In this case, you would use a custom token as you would use the `$Dreamweaver`, `$Fireworks`, `$Flash`, `$fonts`, or `$system` token. For example, if your extension contains multiple files that must be installed in C:\program files\trailer, you can use a token tag to define a custom token called airstream; all of the files that use this token are installed in C:\program files\trailer. If you want to change the destination folder of the files using the `$airstream` token, you have to make only one change in the token tag rather than change every instance of the path to the new destination in your MXI file.

Note: You cannot redefine the `$Dreamweaver`, `$Fireworks`, `$Flash`, `$fonts`, or `$system` token with a custom token.

Attributes

`name`, `{prompt}`, `{default}`, `{definition}`

`name` The name of your custom token. This must be a unique name. Do not include the dollar sign (`$`) in the name.

`prompt` Describes the kind of file to be installed in a folder. When you include this attribute, the user is prompted to specify a destination, and the value you provide is added to the dialog box's title. For example, if the attribute is `prompt="Sample Files"`, the dialog box displays "Select Folder for Sample Files".

`default` Defines the default folder path if the `prompt` attribute is used. If you do not define the `default` attribute, the path box is blank. You can use a token in this attribute, such as `default ="$Dreamweaver"`.

`definition` Defines the file path of the token when you do not use the `prompt` attribute. This prevents the Select Folder dialog box from appearing, so that the user cannot choose a destination path. In the example below, all files using the token `$airstream` are installed in C:\program files\trailer.:

```
<token name="airstream" definition="C:\program files\trailer" />
```

Note: If you use the `prompt` attribute, do not use the `definition` attribute.

Contents

None.

Container

This tag must be contained in a `file-token` tag.

Example

This example is for Windows platforms, which use backslashes (\) to delimit directories:

```
<token name="airstream" prompt="Sample File"
  default="$Dreamweaver\Configuration\Shared\trailer" />
```

This example is for Mac OS, which uses colons (:) to delimit directories:

```
<token name="airstream" prompt="Sample File"
  default="$Dreamweaver:Configuration:Shared:trailer" />
```

Tags and their compatible products

The following table lists each tag and its compatible Adobe products. An X in the application column indicates that the tag is compatible with that product. If there is no X in the column for that product, then the tag is not compatible with the application. If a tag is not supported by an application, but is included in the MXI file, it will be ignored during installation of the extension

Tag	Dreamweaver CS4	Fireworks CS4	Flash CS4
macromedia-extension	X	X	X
description	X	X	X
license-agreement	X	X	X
ui-access	X	X	X
products	X	X	X
product	X	X	X
author	X	X	X
files	X	X	X
file	X	X	X
configuration-changes	X		
documenttype-changes	X		
documenttype-insert	X		
documenttype-remove	X		
toolpanel-changes			X
toolpanel-item-insert			X
ftp-extension-map-changes	X		
ftp-extension-insert	X		
ftp-extension-remove	X		
insertbar-changes	X		
insertbar-insert	X		
insertbar-remove	X		
insertbar-item-insert	X		
insertbar-item-remove	X		
server-behavior-changes	X		
server-format-changes	X		
server-format-definition-changes	X		
data-source-changes	X		
menu-insert	X		

Tag	Dreamweaver CS4	Fireworks CS4	Flash CS4
menu-insert	X		
menu	X		
menu-insert	X		
format	X		
separator	X		
comment	X		
shortcut-remove	X		
shortcut-insert	X		
shortcutlist	X		
shortcut	X		
taglibrary-changes	X		
taglibrary-insert	X		
taglibrary-remove	X		
toolbar-changes	X		
toolbar-insert	X		
toolbar-remove	X		
toolbar-item-insert	X		
toolbar-item-remove	X		
extensions-changes	X		
extension-insert	X		
extension-insert	X		
file-tokens	X	X	X
token	X	X	X

Example MXI file

The following is an example of an MXI file that creates a Dreamweaver extension called Dog and Cat Extension Suite. This particular extension installs an object and a command, and modifies `menu.xml` and `insertbar.xml` to add menu items to the application’s interface.

Reviewers: Please review my addition of the minVersion and maxVersion attributes below. Would it be better to omit these, or include them in only one <file source> tag rather than multiple?

```
<macromedia-extension
  name="Dog and Cat Extension Suite"
  version="1.0.1"
  mxiversion="2.0"
  xmanversion=”2.0”
  icon="icon.png"
  requires-restart="true"
  type="suite">

  <author name="Macromedia" />

  <products>
    <product name="Dreamweaver" version="6" primary="true" />
  </products>
  <description href="http://www.diamond.com/calendarobject/description.htm"
    source="com.diamond/calendarobject/description.htm">
    <![CDATA[This extension installs an object and a command while modifying
    menus.xml and insertbar.xml.]]>
  </description>
  <ui-access>
    <![CDATA[Access the Dog command by selecting Commands > Dog. Access the
      Cat object by selecting the Cat category within the Insert bar.]]>
  </ui-access>
  <license-agreement>
    <![CDATA[SAMPLE THIRD PARTY LICENSE TEXT:<br>
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy
    nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut
    wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit
    lobortis nisl ut aliquip ex ea commodo consequat.]]>
  </license-agreement>
  <files>
    <file source="cat.htm"
      destination="$dreamweaver/configuration/objects/cat"/>
    <file source="cat.gif"
      destination="$dreamweaver/configuration/objects/cat"/>
    <file source="description.htm"
      destination="$ExtensionSpecificEMStore/com.diamond/calendarobject/
  description.htm"/>
    <file source="icon.png"
      destination="$ExtensionSpecificEMStore/icon.png"/>
    <file source="dog.htm"
      destination="$dreamweaver/configuration/commands"/>
    <file source="dog.js"
      destination="$dreamweaver/configuration/commands"/>
  </files>
  <configuration-changes>
    <menu-insert insertAfter="DWMenu_Commands_SortTable"
      skipSeparator="true">
      <menu id="DWMenu_Commands_Dog" name="_Dog">
      </menu>
    </menu-insert>
    <menu-insert appendTo="Animals_Menu_Dog">
      <menuitem id="DWMenu_Commands_Dog_Dog1" name="_Insert Dog" />
      <menuitem id="DWMenu_Commands_Dog_Dog2" name="Insert _Dog Again" />
      <menuitem id="DWMenu_Commands_Dog_Dog3" name="Insert Dog _Again Again"
        file="commands dog.htm" />
    </menu-insert>
    <insertbar-changes>
      <insertbar-insert>
        <category folder="Cat" id="DW_Inserbar_Cat">
          <button file="cat/cat.htm" id="DW_Inserbar_Cat_Cat1" image="cat
            cat.gif" />
        </category>
      </insertbar-insert>
    </insertbar-changes>
  </configuration-changes>
</macromedia-extension>
```

Creating multilingual extension installers (version 2.1 only)

Version 2.1 of Extension Manager supports multilingual extension installers, which let you combine multiple language versions in one MXP file. During installation, the appropriate language is determined by the process outlined in "defaultLanguage" on page 6. Then, language-specific files and text strings are identified using attributes you've included in the MXI file.

Enabling multilingual support in an MXI file

To enable multilingual support in an MXI file, you need to specify the attribute `ismultilingual="true"` to load strings from external files. If you want to localize the name of the extension, you can specify a `name_resid` attribute in the `<macromedia-extension>` tag. If no corresponding language tag is found for the id, "name_ID", Extension Manager falls back to the name in the name attribute.

```
<macromedia-extension
  name="Extension Name"
  name_resid="name_ID"
  version="1.0.0"
  ismultilingual="true" >

<defaultLanguage>en_US</defaultLanguage>
<author name="FooBar" author_resid="author_ID"/>
<description resid="description_ID">
<![CDATA[ Inline Description ]]>
</description>
```

Setting up the folder hierarchy for localized XML files

At the same location as your .mxi file, create a folder with the exact name of your .mxi file and append "Resources" to it. For example if you have an .mxi file named "Calendar.mxi", create a folder named "Calendar.mxi_Resources". In this folder, you add an XML file for each language you are localizing the extension into. Each file will have a two-letter ISO language code followed by an underscore character, and then the two-letter ISO country code in upper case. For example, for English you specify "en_US.xml" and for French "fr_FR.xml".

If your extension needs to install localized files, you may want to create a subfolder for each of the languages. The folder hierarchy should look as follows:

Calendar.mxi - MXI File

Calendar.mxi_Resources (folder)

en_US.xml - XML File containing English strings

fr_FR.xml - XML File containing French Strings

en_US (Folder Containing English files)

fr_FR (Folder Continaing French files)

Formatting XML for language-specific strings

The Extension Manager looks up localized strings from XML files that you provide for each language. Each XML file should use Adobe's zstring format. Below is an example of the French xml file. The locale (in this case, `"fr_FR")` should match the locale of the language. Extension Manager looks up the localized strings based on the `name=""` attribute and subsitutes the localized strings in the `<val>` tags. In the following example, we use `name_ID` for the localized Extension Name, "French Extension Name". And `description_ID` will be used for the Description displayed when you click the Extension in the Extension Manager.

Example

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE asf SYSTEM "http://ns.adobe.com/asf/asf_1_0.dtd">
<asf locale="fr_FR" version="1.0" xmlns="http://ns.adobe.com/asf">
  <set name="DefaultSet">
    <str name="name_ID">
      <val>French Extension Name</var>
    </str>
    <str name="description_ID">
      <val>
        French Extension Description.
</val> </str> </set> </asf>
```

Installing localized files

To specify that a set of localized files get installed for a particular language, use the `xml:lang` attribute on the `files` tag containing files for that language.

Example

```
<files xml:lang="en_US">
  <file source="en_US/Jacket.htm" destination="$dreamweaver/configuration"/>
  </files>

  <files xml:lang="fr_FR">
  <file source="fr_FR/Jacket.htm" destination="$dreamweaver/configuration"/>
  </files>

  <files>
  <file source="styles.css" destination="$dreamweaver/configuration" />
  </files>
```

Configuration changes

For Dreamweaver, you can provide localized strings for changes to the XML files in the `<configuration-changes>` section of the mxi file by using the `resid:attributeName` and providing a string id in the string resource files. For example, to provide a localized menu item string, you add `resid:name="menuItem_ID"` to the `<menuitem>` tag.

Example

```
<configuration-changes>
  <menu-insert appendTo="DWMenu_Insert">
  <menu id="MyItem_Insert" name="Menu Item Default" resid:name="menuItem_ID" >
  </menu-insert>
</configuration-changes>
```

If no localized string is available for the language being installed, the default string in the name= attribute "Menu Item Default" is inserted. Using this method, you can provide localized strings for the following attributes:

Menu Items: `<menuitem resid:name="menuItem_ID" />`

Button Names: `<button resid:label="buttonLabel_ID" />`

Formatting command-line arguments

Parameter definitions

- "-install" installs the extension
- "-package" packages the extension
- "-remove" removes the extension
- "-enable" enables the extension
- "-disable" disables the extension
- "-locate" locates point product in Extension Manager window
- "-suppress" suppresses the EULA dialog and “Installation succeeded” confirmation
- "-locale" specifies language for Extension Manager at startup
- "-EMBT" used only when command line parameter is passed through BridgeTalk. "-EMBT" should appear before all other command line parameters.

Parameter formats

Required parameter	Following parameters allowed
-install	mxi="", mxp=""
-package	mxi="", mxp=""
-remove	product="", extension=""
-enable	product="", extension=""
-disable	product="" extension=""
-locate	product=""
-locale lang=""	-locate, -install, -package, -remove, -enable, -disable
-suppress	-locate, -install, -package, -remove, -enable, -disable
-EMBT	any parameter above

Examples:

Install extension

- Windows: "Adobe Extension Manager CS4.exe" -install mxp="d:\test.mxp"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -install mxp="/Volumns/x1/test.mxp"

"Package extension

- Windows: "Adobe Extension Manager CS4.exe" -package mxi="d:\test.mxi" mxp="d:\test.mxp"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -package mxi="="/Volumns/x1/test.mxi" mxp="/Volumns/x1/test.mxp"

"Remove extension

- Windows: "Adobe Extension Manager CS4.exe" -remove product="Dreamweaver CS4" extension="Sample"
  "Adobe Extension Manager CS4.exe" -remove productfamily="Photoshop-11" extension="Sample"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -remove product="Dreamweaver CS4" extension="Sample"

"Enable extension

- Windows: "Adobe Extension Manager CS4.exe" -enable product="Dreamweaver CS4" extension="Sample"
  "Adobe Extension Manager CS4.exe" -enable productfamily="Photoshop-11" extension="Sample"

- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -enable product="Dreamweaver CS4" extension="Sample"

"Disable extension

- Windows: "Adobe Extension Manager CS4.exe" -disable product="Dreamweaver CS4" extension="Sample"
  "Adobe Extension Manager CS4.exe" -disable productfamily="Phtoshop-11" extension="Sample"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -disable product="Dreamweaver CS4" extension="Sample"

"Locate the point product in Extension Manager window

- Windows: "Adobe Extension Manager CS4.exe" -locate product="Dreamweaver CS4"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -locate product="Dreamweaver CS4"

"Install extension and suppress EULA dialog and “Installation succeeded” confirmation

- Windows: "Adobe Extension Manager CS4.exe" -suppress -install mxp="d:\test.mxp"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -suppress -install mxp="/Volumns/x1/test.mxp"

"Specify en_US as language at startup

- Windows: "Adobe Extension Manager CS4.exe" -locale lang="en_US" -install mxp="d:\test.mxp"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -locale lang="en_US" -install mxp="/Volumns/x1/test.mxp"

"Parameters through BridgeTalk

First launch through BridgeTalk:

- Windows: "Adobe Extension Manager CS4.exe" -EMBT -locale lang="en_US" -install mxp="d:\\test.mxp"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -EMBT -locale lang="en_US" -install mxp="/Volumns/x1/test.mxp"

Install through BridgeTalk:

- Windows: "Adobe Extension Manager CS4.exe" -EMBT -install mxp="d:\test.mxp"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -EMBT -install mxp="/Volumns/x1/test.mxp"

Package through BridgeTalk

- Windows: "Adobe Extension Manager CS4.exe" -EMBT -package mxi="d:\test.mxi" mxp="d:\test.mxp"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -EMBT -package mxi="/Volumns/x1/test.mxi" mxp="/Volumns/x1/test.mxp"

Remove through BridgeTalk

- Windows: "Adobe Extension Manager CS4.exe" -EMBT -remove product="Dreamweaver CS4" extension="Sample"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -EMBT -remove product="Dreamweaver CS4" extension="Sample"

Enable through BridgeTalk

- Windows: "Adobe Extension Manager CS4.exe" -EMBT -enable product="Dreamweaver CS4" extension="Sample"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -EMBT -enable product="Dreamweaver CS4" extension="Sample"

Disable through BridgeTalk

- Windows: "Adobe Extension Manager CS4.exe" -EMBT -disable product="Dreamweaver CS4" extension="Sample"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -EMBT -disable product="Dreamweaver CS4" extension="Sample"

Locate through BridgeTalk

- Windows: "Adobe Extension Manager CS4.exe" -EMBT -locate product="Dreamweaver CS4"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -EMBT -locate product="Dreamweaver CS4"

Suppress through BridgeTalk

- Windows: "Adobe Extension Manager CS4.exe" -EMBT -suppress -install mxp="d:\test.mxp"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -EMBT -suppress -install mxp="/Volumns/x1/test.mxp"

Invalid parameters

- Five parameters (mxi, mxp, product, extension, lang) do not allow "-" at the front of them.
- Incorrect parameter spelling will show the help message for the command-line interface in the console (except for the seven required parameters listed in the table).
- -locale parameter must be followed by lang="en_US", otherwise it is a invalid parameter.

Valid irregular parameters (which support dragging or double-clicking to install and package)

Install

- Windows: "Adobe Extension Manager CS4.exe" "d:\test.mxp"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" "/Volumns/x1/test.mxp"

Package

- Windows: "Adobe Extension Manager CS4.exe" "d:\\test.mxi"
  "Adobe Extension Manager CS4.exe" -suppress "d:\\test.mxi"
- Mac OS: "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" "/Volumns/x1/test.mxi"
  "/Applications/Adobe Extension Manager CS4/Adobe Extension Manager CS4.app/Contents/MacOS/Adobe Extension Manager CS4" -suppress "/Volumns/x1/test.mxi"

"If a string (for example, extension name, or mxi/mxp file name) does not contain a space character, it does not need to be included in ""

For example, both below are valid:

- -enable product="Flash CS4" extension=Sample
- -enable product="Flash CS4" extension="Sample"