WixUI dialog library

The WixToolset.UI.wixext WiX extension offers several sets of dialogs using Windows Installer internal UI.

Dialog sets

The WixUI dialog library contains the following built-in dialog sets that provide a familiar wizard-style setup user interface.

WixUI id	Description
WixUI_Advanced	The WixUI_Advanced dialog set provides the option of a one-click install like WixUI_Minimal, but it also allows directory and feature selection like other dialog sets if the user chooses to configure advanced options.
WixUI_FeatureTree	WixUI_FeatureTree is a simpler version of WixUI_Mondo that omits the setup type dialog. Instead, the wizard proceeds directly from the license agreement dialog to the feature customization dialog. WixUI_FeatureTree is more appropriate than WixUI_Mondo when your product installs all features by default.
WixUI_InstallDir	WixUI_InstallDir does not allow the user to choose what features to install, but it adds a dialog to let the user choose a directory where the entire product will be installed.

WixUI id	Description
WixUI_Minimal	WixUI_Minimal is the simplest of the built-in WixUI dialog sets. Its sole dialog combines the welcome and license agreement dialogs and omits the feature customization dialog. WixUI_Minimal is appropriate when your product has no optional features and does not support changing the installation directory.
WixUI_Mondo	WixUI_Mondo includes a set dialogs that allow granular installation customization options. WixUI_Mondo is appropriate when some product features are not installed by default and there is a meaningful difference between typical and complete installs. Note: WixUI_Mondo uses SetInstallLevel control events to set the install level when the user chooses *Typical* or *Complete*. For *Typical*, the install level is set to 3; for *Complete*, 1000. For details about feature levels and install levels, see INSTALLLEVEL Property.

Adding a WixUI dialog set to your MSI package

To add a WixUI dialog set to your MSI package:

1. Add a reference to the WixToolset.UI.wixext WiX extension.
2. Add the WixToolset.UI.wixext WiX extension namespace to your WiX authoring.
3. Add the `WixUI` element to your WiX authoring.

For example:

```
<Wix xmlns="http://wixtoolset.org/schemas/v4/wxs"
  xmlns:ui="http://wixtoolset.org/schemas/v4/wxs/ui">

  <Package ...>
      ...
      <ui:WixUI
        Id="WixUI_InstallDir"
        InstallDirectory="INSTALLFOLDER"
        />
  </Package>
</Wix>
```

Localization

Translated strings

Translated strings are included in WixToolset.UI.wixext for the following cultures:

- ar-SA
- bg-BG
- ca-ES
- cs-CZ
- da-DK
- de-DE
- el-GR
- en-US
- es-ES
- et-EE
- fi-FI

- fr-FR
- he-IL
- hi-IN
- hr-HR
- hu-HU
- it-IT
- ja-JP
- kk-KZ
- ko-KR
- lt-LT
- lv-LV
- nb-NO
- nl-NL
- pl-PL
- pt-BR
- pt-PT
- ro-RO
- ru-RU
- sk-SK
- sl-SI
- sq-AL
- sr-Latn-RS
- sv-SE

- th-TH
- tr-TR
- uk-UA
- zh-CN
- zh-HK
- zh-TW

Using translated error and progress text

By default, WixToolset.UI.wixext does not include any translated Error or ProgressText elements in your packages. To include them, reference the WixUI_ErrorProgressText UI element:

```
<UIRef Id="WixUI_ErrorProgressText" />
```

Building localized packages that use WixUI

To build a localized package using the Wix.exe command-line tool:

- Add a `-culture` switch with an argument of the culture you want to build. The localized strings for that culture in WixToolset.UI.wixext (if available) will be used.

To build a localized package using a .wixproj MSBuild project:

- Set the `Cultures` property in the .wixproj to a semicolon-delimited list of cultures. For example:

```xml
<PropertyGroup>
  <Cultures>en-US;ja-JP</Cultures>
</PropertyGroup>
```

WixToolset.UI.wixext includes a placeholder license agreement. To specify your product's license, override the placeholder by specifying a WiX variable named `WixUILicenseRtf` with the value of a path to an RTF file that contains your license text. You can define the variable in your WiX authoring:

```xml
<WixVariable
  Id="WixUILicenseRtf"
  Value="bpl.rtf"
  />
```

Alternatively, you can define the variable using the `-bindvariable` switch when running Wix.exe:

```
wix build -ext WixToolset.UI.wixext -bindvariable WixUILicenseRtf=bpl.rtf Product.wxs -out Product.msi
```

Or when using MSBuild, you can add a `BindVariable` item:

```xml
<ItemGroup>
  <BindVariable Include="WixUILicenseRtf=bpl.rtf" />
</ItemGroup>
```

TIP

There is a known issue on some (typically older) versions of Windows with the rich text control used to display the text of the license file. The text can appear blank until the user scrolls down in the control. This is typically caused by complex RTF content (such as the RTF generated when saving an RTF file in Microsoft Word). If you run into this behavior in your setup UI, one of the following workarounds will fix it in most cases:

- Open your RTF file in WordPad and save it from there in order to remove the complex RTF content from the file. After saving it, rebuild your MSI.
- Use a dialog set that has a "welcome" page before showing the license. This problem typically only occurs when the license agreement screen is the first one displayed during setup.

Replacing the default bitmaps

The WixUI dialog library includes default bitmaps for the background of the welcome and completion dialogs and the top banner of the other dialogs. You can replace those bitmaps with your own for product branding purposes. To replace default bitmaps, specify WiX variable values with paths to your bitmaps, just like when replacing the default license text.

Variable name	Description	Dimensions
WixUIBannerBmp	Top banner	493 × 58
WixUIDialogBmp	Background bitmap used on the welcome and completion dialogs	493 × 312
WixUIExclamationIco	Exclamation icon on the WaitForCostingDlg	32 × 32
WixUIInfoIco	Information icon on the cancel and error dialogs	32 × 32

Variable name	Description	Dimensions
WixUINewIco	Button glyph on the BrowseDlg	16 × 16
WixUIUpIco	Button glyph on the BrowseDlg	16 × 16

```
<ui:WixUI
  Id="WixUI_FeatureTree"
  />
<WixVariable
  Id="WixUIDialogBmp"
  Value="WixUIDialogBmp.png"
  />
<WixVariable
  Id="WixUIBannerBmp"
  Value="WixUIBannerBmp.png"
  />
```

Note that on modern versions of Windows you can use modern graphics file formats, like PNG.

Adding text or a checkbox to the completion dialog

To show optional text on the completion dialog (ExitDlg), set the WIXUI_EXITDIALOGOPTIONALTEXT property to the string you want to show. For example:

```
<Property
  Id="WIXUI_EXITDIALOGOPTIONALTEXT"
  Value="Thank you for installing this product."
  />
```

Likewise, to show an optional checkbox on ExitDlg, set the WIXUI_EXITDIALOGOPTIONALCHECKBOXTEXT property to the string you want to show. For example:

```
<Property
  Id="WIXUI_EXITDIALOGOPTIONALCHECKBOXTEXT"
  Value="Launch My Application Name"
  />
```

The optional text and checkbox have the following behaviors:

- Long strings will wrap across multiple lines.
- The optional text is only shown during initial installation, not during maintenance mode or uninstall.
- The optional text is displayed as literal text, so properties surrounded by square brackets, like `[ProductName]` will not be resolved. If you need to include property values in the optional text, you must schedule a custom action to set the property value. For example:

```
<SetProperty
  Id="WIXUI_EXITDIALOGOPTIONALTEXT"
  Value="Thank you for installing [ProductName]."
  After="FindRelatedProducts"
  Sequence="ui"
  />
```

Changing the text of a control

All the text in WixUI dialogs is supplied by localization strings embedded in WixToolset.UI.wixext. Each of those strings is marked `Overridable="yes"`, which means you can replace them in your own package's .wxl file. For example, to override the descriptive text on WelcomeDlg, you would add the following to a .wxl file in your project:

```
<String
  Id="WelcomeDlgDescription"
  Value="This is a custom welcome message. Click Next to continue or Cancel to exit."
  />
```

Adding and removing dialogs from a dialog set

Each WixUI dialog set is a wizard-style sequence of dialogs wired up to Next and Back buttons. To remove a dialog from the wizard sequence, you have to adjust the control events on the Next and Back buttons to point "beyond" the dialog you want to skip. To add a new dialog, you have to modify existing control events and add new ones on the Next and Back buttons to point to the new dialog you want to add.

In general, you'll need to duplicate the original WiX authoring for the dialog set you want to modify. You can see the WiX authoring in the `wix4` repo on GitHub.

For example, to remove LicenseAgreementDlg from the WixUI_InstallDir dialog set:

1. Copy WixUI_InstallDir.wxs in the WiX source code to your project.
2. Rename the `UI` element `Id`s. For example:

```
<UI Id="InstallDir_NoLicense_$(WIXUIARCH)">
...
```

```
<UI Id="file InstallDir_NoLicense">
```

3. Remove the `Publish` elements that are used to add Back, Next, and Print events for LicenseAgreementDlg.
4. Change the `Publish` element that is used to add a Next event to WelcomeDlg to go to InstallDirDlg instead of LicenseAgreementDlg. For example:

```
<Publish
  Dialog="WelcomeDlg"
  Control="Next"
  Event="NewDialog"
  Value="InstallDirDlg"
  />
```

5. Change the `Publish` element that is used to add a Back event to InstallDirDlg to go to WelcomeDlg instead of LicenseAgreementDlg. For example:

```
<Publish
  Dialog="InstallDirDlg"
  Control="Back"
  Event="NewDialog"
  Value="WelcomeDlg"
  />
```

Likewise, to add a dialog named SpecialDlg between WelcomeDlg and LicenseAgreementDlg in the WixUI_InstallDir dialog set:

1. Define the appearance of SpecialDlg in a `UI` element in your project.
2. Copy WixUI_InstallDir.wxs in the WiX source code to your project.
3. Rename the `UI` element `Id`s. For example:

4.
```
<UI Id="InstallDir_SpecialDlg_$(WIXUIARCH)">
```

```
<Publish
  Dialog="SpecialDlg"
  Control="Back"
  Event="NewDialog"
  Value="WelcomeDlg"
  />
<Publish
  Dialog="SpecialDlg"
  Control="Next"
  Event="NewDialog"
  Value="LicenseAgreementDlg"
  />
```

5. Change the `Publish` element that is used to add a Next event to WelcomeDlg to go to SpecialDlg instead of LicenseAgreementDlg. For example:

```
<Publish
  Dialog="WelcomeDlg"
  Control="Next"
  Event="NewDialog"
  Value="SpecialDlg"
  />
```

6. Change the `Publish` element that is used to add a Back event to LicenseAgreementDlg to go to SpecialDlg instead of WelcomeDlg. For example:

```
<Publish
```

```
    Dialog="LicenseAgreementDlg"
    Control="Back"
    Event="NewDialog"
    Value="SpecialDlg"
    />
```

Edit this page