Using And Controlling The RadioButton/RadioButtonGroup Controls In IPWI *(and their associated private Windows Installer Properties).*

Abstract:

Whenever we have the requirement to author an installation at a more advanced level, this invariably involves us modifying and customizing the user interface that is displayed to the end user.

Of the many standard controls available to Windows Installer are the RadioButton and RadioButtonGroup controls which work hand-in-hand to allow the setup author to present the end user with options and perform processing based on those choices.

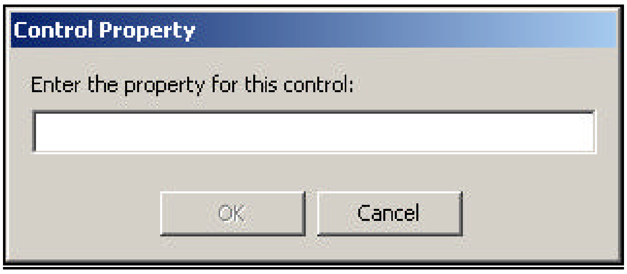
This article will expose the workings of the aforementioned controls and give some technical advice as how to really determine how these work, expose some of the 'hidden' and undocumented functionality of Windows Installer properties which work around these controls and finally present some practical and 'real world' scenarios where you might want to do this.

How the RadioButtonGroup Control Works.

In order to achieve the 'usual' behavior that we are used to for a RadioButton control, that is to say that there is a 'group' of RadioButtons and they are being used to set some value (in our case a Windows Installer Property) based on which one was selected, we must first apply a RadioButtonGroup control to the dialog.

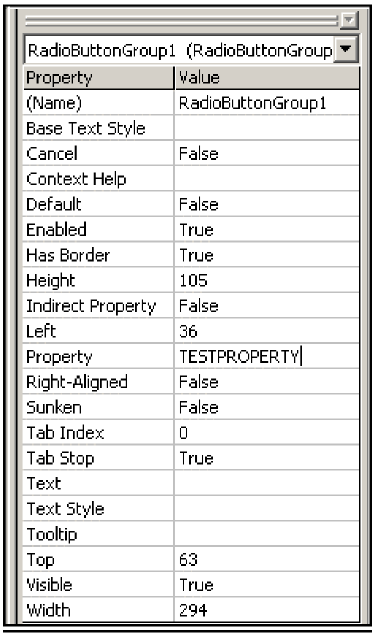
Once you add a RadioButtonGroup to a dialog using the toolbar button, IPWI will prompt you to enter a property which will represent that RadioButtonGroup. We will use this RadioButtonGroup control as a placeholder for RadioButton controls, and it will be these RadioButton controls which set the Property (specified for our RadioButtonGroup) to the appropriate value, based on the option the end-user selected.

Below can be seen the dialog requesting the name of the property for the RadioButtonGroup:



We see with other dialog controls, the RadioButtonGroup control has many properties pertaining to the look and feel we can apply to the control. However, the most important property of this RadioButtonGroup control is the 'Property' property. It is this property item which holds the actual Windows Installer Property that the RadioButtonGroup will be exposing to the RadioButton controls within it *(covered in the next section).*

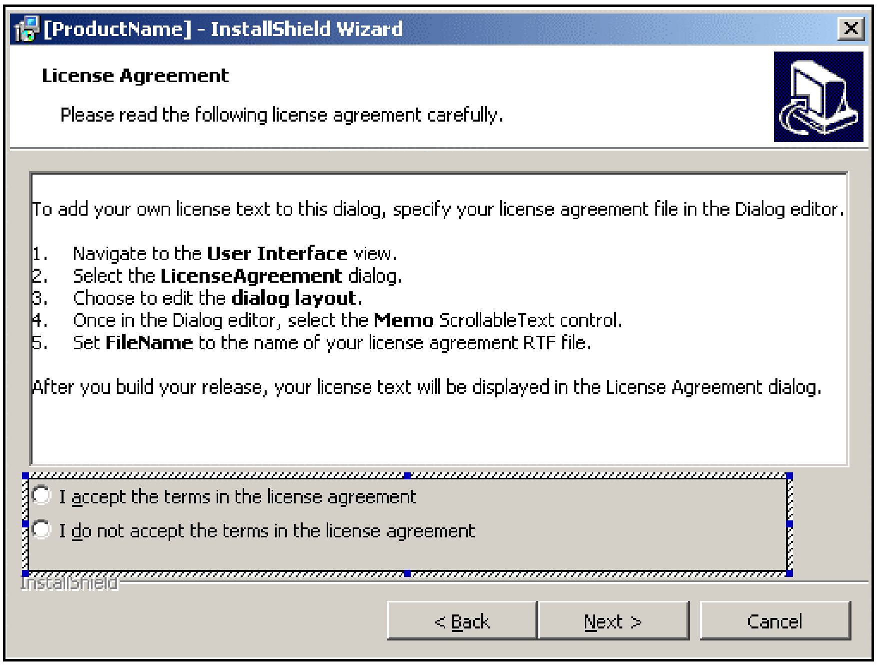
Once added, your RadioButtonGroup will have a 'framed' border visible by default, this is represented by the 'Has Border' property of the control and can be set to either 'True' or '. We can see the properties list below, showing the 'Property' property mentioned above *in this case set to 'TESTPROPERTY')* and the recently mentioned 'Has Border' property:



RadioButtonGroup1 (RadioButtonGroup

Property	Value
(Name)	RadioButtonGroup1
Base Text Style	
Cancel	False
Context Help	
Default	False
Enabled	True
Has Border	True
Height	105
Indirect Property	False
Left	36
Property	TESTPROPERTY
Right-Aligned	False
Sunken	False
Tab Index	0
Tab Stop	True
Text	
Text Style	
Tooltip	
Top	63
Visible	True
Width	294

If 'Has Border' is set to 'False' the control will have no border much like the standard dialog we see below for the LicenseAgreement (the only reason we see a border here is because it has been selected for illustration purposes):


[ProductName] - InstallShield Wizard
License Agreement
Please read the following license agreement carefully.
InstallShield
< Back
Next >
Cancel

To add your own license text to this dialog, specify your license agreement file in the Dialog editor.

1. Navigate to the **User Interface** view.
2. Select the **LicenseAgreement** dialog.
3. Choose to edit the **dialog layout**.
4. Once in the Dialog editor, select the **Memo** ScrollableText control.
5. Set **FileName** to the name of your license agreement RTF file.

After you build your release, your license text will be displayed in the License Agreement dialog.

○ I accept the terms in the license agreement

○ I do not accept the terms in the license agreement

Things to note:

- You MUST first place a RadioButtonGroup onto your Dialog if you want to use RadioButton controls, this will NOT be done automatically for you.
- When you delete a RadioButtonGroup, all of the RadioButton controls within it will also be removed.
- Windows Installer handles the whole group as one single control, thus, individual RadioButton controls cannot be independently enabled/disabled.
- When manipulating the user interface of a dialog the easiest and best way to find the RadioButtonGroup control is to select it from the pull-down list at the top of the properties list we saw on the previous page.

As with all Dialog controls, the RadioButtonGroup control is stored in the *Control* table, to quickly associate this control with a Property (user-defined or otherwise) you can populate the 'Property' Column of this *Control* table.

Once you have created a RadioButtonGroup, you can go ahead and drop some RadioButton controls onto it – this is covered in the next section.

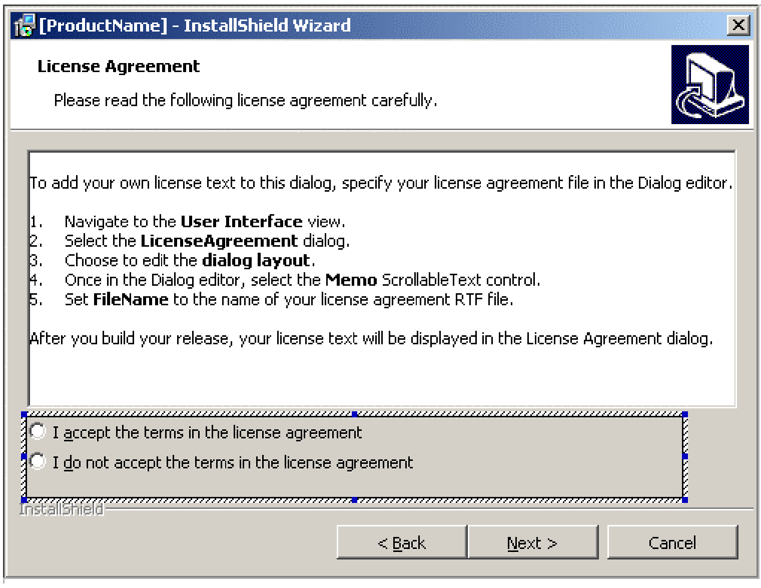
How the RadioButton Control Works.

Before we go into this topic we must first be clear about what Windows Installer thinks of as a RadioButton Control and ensure that we appreciate the difference between this and a 'normal' Windows - style RadioButton control.

In Windows, a RadioButton control will work independently, whereas a Windows Installer RadioButton Control is not seen as an individual control, but always as part of a group; using the RadioButtonGroup we discussed earlier. As such, the whole RadioButtonGroup will operate as one control, (regardless of the number of RadioButton Controls within it), changing the value of the RadioButtonGroup's Property' property based on the individual RadioButton that the end-user selected.

To add a RadioButton Control to your dialog, simply select the RadioButton Control toolbar button and drop it onto a previously created RadioButtonGroup.

If we take a dialog such as the LicenseAgreement Dialog we firstly see it has a RadioButtonGroup called 'Agree', as shown below:



[ProductName] - InstallShield Wizard

License Agreement

Please read the following license agreement carefully.

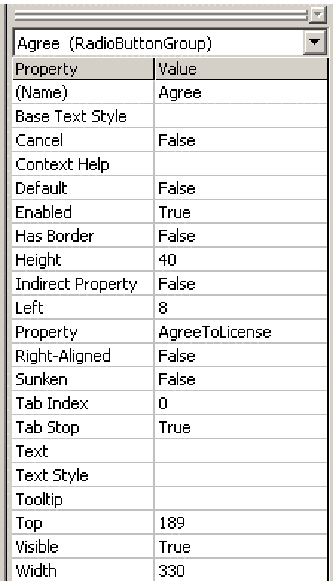
To add your own license text to this dialog, specify your license agreement file in the Dialog editor.

1. Navigate to the **User Interface** view.
2. Select the **LicenseAgreement** dialog.
3. Choose to edit the **dialog layout**.
4. Once in the Dialog editor, select the **Memo** ScrollableText control.
5. Set **FileName** to the name of your license agreement RTF file.

After you build your release, your license text will be displayed in the License Agreement dialog.

○ I accept the terms in the license agreement

○ I do not accept the terms in the license agreement



Agree (RadioButtonGroup)

Property	Value
(Name)	Agree
Base Text Style	
Cancel	False
Context Help	
Default	False
Enabled	True
Has Border	False
Height	40
Indirect Property	False
Left	8
Property	AgreeToLicense
Right-Aligned	False
Sunken	False
Tab Index	0
Tab Stop	True
Text	
Text Style	
Tooltip	
Top	189
Visible	True
Width	330

We see that the RadioButtonGroup has its 'Property' property set to 'AgreeToLicense'. This is the property which applies to all the RadioButton Controls that are placed within this RadioButtonGroup.

You will notice that from the screenshot above, the LicenseAgreement Dialog has two RadioButtonControls within the RadioButtonGroup. If we inspect the properties list for each of these RadioButton Controls we will see that they have a property called 'Va lue'.

The 'Property' property of the RadioButtonGroup is set to the 'Value' property of the RadioButton Control when it is selected by the end-user. Taking a look at the default functionality of a Windows Installer Installation, we see that the individal RadioButton Controls on the Dialog above have the following settings for their respective 'Value' properties:

AgreeToLicense1 ("I accept the terms of the license…")	Yes
AgreeToLicense2 ("I do not accept the terms of the license…")	No

As a result, once the end-user selects either 'I accept the terms of the license…', or 'I do not accept the terms of the license' the property 'AgreeToLicense is set to either 'Yes' or 'No'.

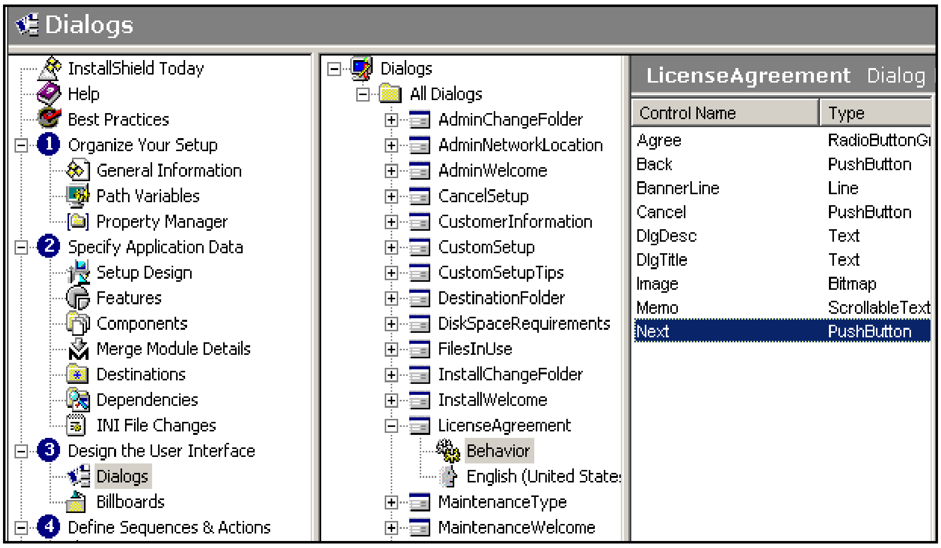
If you want to make an RadioButton selected as default on your dialog, you simply need to use the Property Manager (off the 'Organize Your Setup' section of IPWI 2.0x) and make a new entry under the name of the RadioButtonGroup 'Property' property. If you want to ensure the scope of this property is public, make sure the name of the property is in uppercase. Next, enter the value which matches the 'Value' property of the RadioButton control you want to become the default option. This will force the nominated RadioButton control to be selected as default when the Dialog is displayed.

Within the Dialog Editor of IPWI you have two toolbar buttons of interest, the RadioButtonGroup Control; and the RadioButton Control itself; . In Windows Installer there is no real separate RadioButton Control, behind-the-scenes, we create RadioButton Controls by adding entries to the *RadioButton* Table and link these entries to the RadioButtonGroup by means of the RadioButtonGroup 'Property' property. The toolbar button has been provided as an quick method of populating this *RadioButton* Table (See MSI help for more information).

Each RadioButton Control also has a property called 'Order'. This must be a positive integer and represents the ordering of the RadioButtons on the dialog. The default ordering for RadioButton Controls is their order of creation, i.e.- the order in which they were added to the Dialog in the Dialog Editor. *(These do not have to be consecutive numbers).*

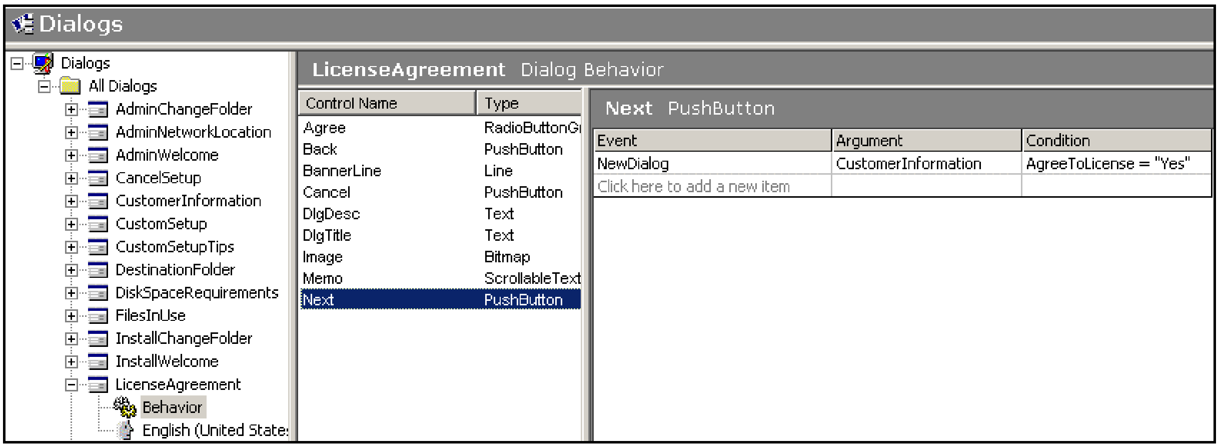
So, now that we have a more in-depth understanding of the RadioButton Control, lets use this information on what the user selected by looking at how this is used to control the installation. When we run an installation, the 'Next' button is not enabled until the end-user selects the 'I accept the terms of the license agreement' option on the 'LicenseAgreement' dialog. This works by using the 'AgreeToLicense' property in a Condition.

When we are using Windows Installer, we have no concept of variables like we might have in a programming language, instead we have 'Properties'. Some of these 'Properties' can be viewed and manipulated via the Property Manager off the 'Organize Your Setup' section. The 'AgreeToLicense' property is used by the 'Next' button on the 'LicenseAgreement' Dialog. To inspect what really goes on here, we must go to the 'Behaviours' section of the Dialog, as shown below, and select the 'Next' PushButton Control:



Control Name	Type
Agree	RadioButtonGi
Back	PushButton
BannerLine	Line
Cancel	PushButton
DlgDesc	Text
DlgTitle	Text
Image	Bitmap
Memo	ScrollableText
Next	PushButton

Once we have selected the 'Next' PushButton Control we can see that it fires a ControlEvent called 'NewDialog' whic h informs the installer to move on from one modal dialog box to another, thus creating the 'flow' of dialogs so typical of installations. The 'Argument' parameter shown is the name of the dialog to display, in the example below, the next dialog to be show n is the 'CustomerInformation' dialog:



Control Name	Type
Agree	RadioButtonGı
Back	PushButton
BannerLine	Line
Cancel	PushButton
DlgDesc	Text
DlgTitle	Text
Image	Bitmap
Memo	ScrollableText
Next	PushButton

Event	Argument	Condition
NewDialog	CustomerInformation	AgreeToLicense = "Yes"
Click here to add a new item		

On the previous page we were discussing the use of the 'AgreeToLicense' property within a standard installation. In the screenshot above we can see that this property can be used within a Condition much like any other Windows Installer property, this example shows how the 'CustomerInformation' Dialog is only displayed (via the NewDialog ControlEvent) if the value of 'AgreeToLicense' equals 'Yes'.

The 'AgreeToLicense' property would only equal 'Yes' if the appropriate RadioButton Control was selected by the end -user, referencing the table we saw earlier in this article we can double-check which individual RadioButton Control was responsible for this:

AgreeToLicense1 ("I accept the terms of the license…")	Yes
AgreeToLicense2 ("I do not accept the terms of the license…")	No

Remembering that the RadioButton 'Value' property updates the RadioButtonGroup 'Property' property, it is this we ar e using in a Condition illustrated in the at the top of this page, and shown below:

AgreeToLicense = "Yes"

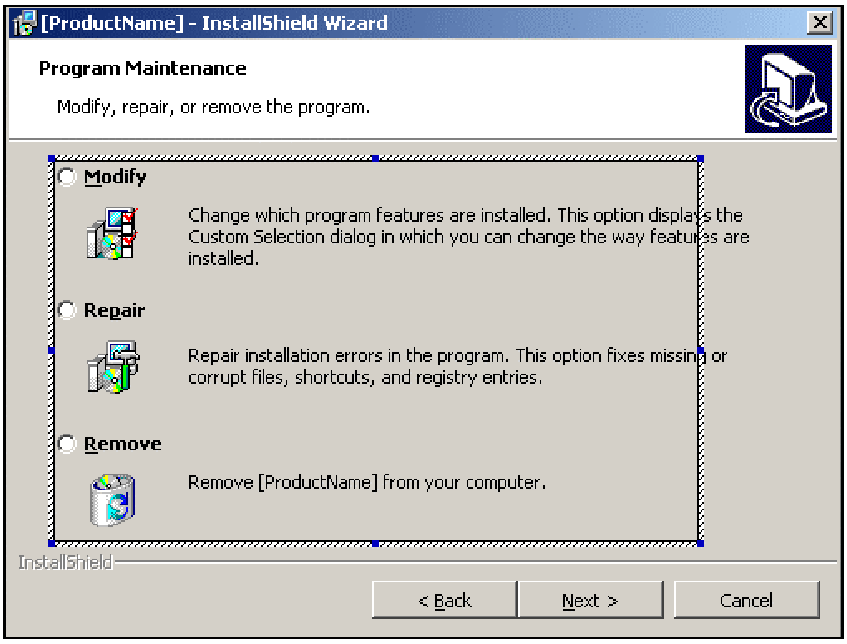
Hidden Windows Installer properties available for use in your projects.

Now that we have looked at how the RadioButtonGroup and RadioButton Controls interact with one another and set Windows Installer Properties, I will take this opportunity to expose some more of the functionality of Windows Installer installations. You may recall that to date we have been using the 'LicenseAgreement' Dialog to illustrate the aforementioned Dialog controls. We also saw how the 'AgreeToLicense' property was set and updated according to the user input via the Dialog's RadioButtons.

Whenever we re -run an installation with Windows Installer it will, by default, invoke what we call the 'Maintenance Mode' for the installation. This will present the end -user with a alternative dialog with three main options, these options are:

- Modify
- Repair
- Remove

These options are represented by RadioButton Controls, which of course are part of a group represented by the RadioButtonGroup control, as per the screenshot below: –



Program Maintenance

Modify, repair, or remove the program.

○ Modify

Change which program features are installed. This option displays the Custom Selection dialog in which you can change the way features are installed.

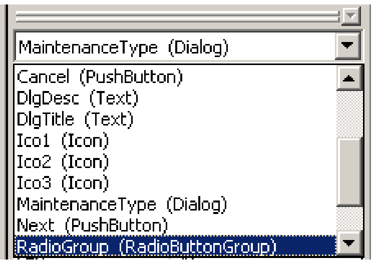
○ Repair

Repair installation errors in the program. This option fixes missing or corrupt files, shortcuts, and registry entries.

○ Remove

Remove [ProductName] from your computer.

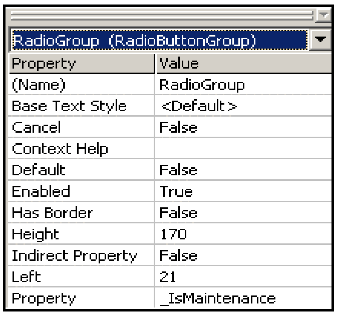
The RadioButtonGroup selected above can be seen from the properties list (below):



MaintenanceType (Dialog)

Cancel (PushButton)
DlgDesc (Text)
DlgTitle (Text)
Ico1 (Icon)
Ico2 (Icon)
Ico3 (Icon)
MaintenanceType (Dialog)
Next (PushButton)
RadioGroup (RadioButtonGroup)

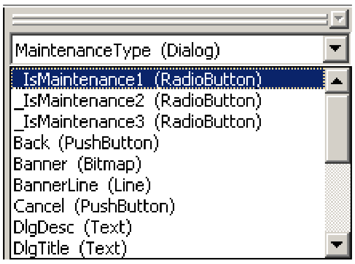
Notice how the RadioButtonGroup named 'RadioGroup' has the 'Property' property populated as '_IsMaintenance';



RadioGroup (RadioButtonGroup)

Property	Value
(Name)	RadioGroup
Base Text Style	<Default>
Cancel	False
Context Help	
Default	False
Enabled	True
Has Border	False
Height	170
Indirect Property	False
Left	21
Property	_IsMaintenance

This '_IsMaintenance' Property is populated with the 'Value' property from any one of the three RadioButton controls visible on the Dialog. The three RadioButtons on this Dialog can be seen in the pull-down list of Dialog Controls below; their respective 'Value' Properties can be seen in the table on the right-hand side:



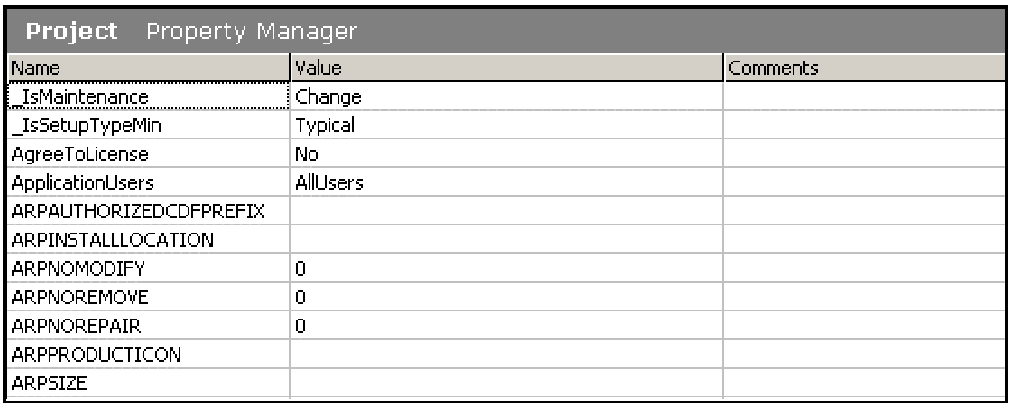
MaintenanceType (Dialog)

_IsMaintenance1 (RadioButton)
_IsMaintenance2 (RadioButton)
_IsMaintenance3 (RadioButton)
Back (PushButton)
Banner (Bitmap)
BannerLine (Line)
Cancel (PushButton)
DlgDesc (Text)
DlgTitle (Text)

[redacted]	[redacted]	[redacted]
_IsMaintenance1	Modify	Change
_IsMaintenance2	Repair	Reinstall
_IsMaintenance3	Remove	Remove

Using the concepts we discussed earlier in this article we can see how by using the Property Manager shown below we can force one of the RadioButton controls to be the default option by changing the value of '_IsMaintenance' to match one of the values illustrated in the table above *(i.e. 'Change','Reinstall' or 'Remove').*

Note that the default value for this '_IsMaintenance' property is 'Change' forcing the 'Modify' *(middle)* RadioButton to be selected when the dialog is first shown:



Project Property Manager

Name	Value	Comments
_IsMaintenance	Change	
_IsSetupTypeMin	Typical	
AgreeToLicense	No	
ApplicationUsers	AllUsers	
ARPAUTHORIZEDCDFPREFIX		
ARPINSTALLLOCATION		
ARPNOMODIFY	0	
ARPNOREMOVE	0	
ARPNOREPAIR	0	
ARPPRODUCTICON		
ARPSIZE		

This '_IsMaintenance' property is a largely un-documented property of Windows Installer, as such you will not find it referenced in the IPWI help or the Microsoft Windows Installer help library. There are articles available at support.installshield.com with more information on the different functionality for each of the possible values of the '_IsMaintenance' property.

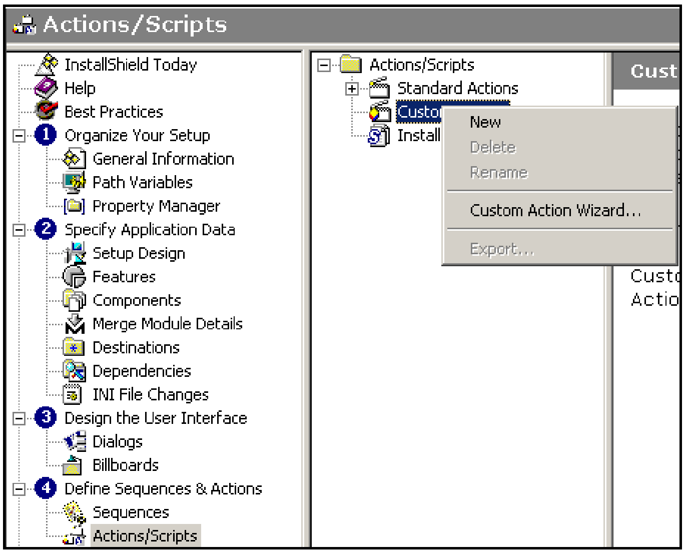
Utililizing these Properties within a Custom Action Condition to achieve a 'real -world' solution.

The final part of this article is to propose a real-world scenario where you might want to use this un - documented '_IsMaintenance' Property to your advantage.

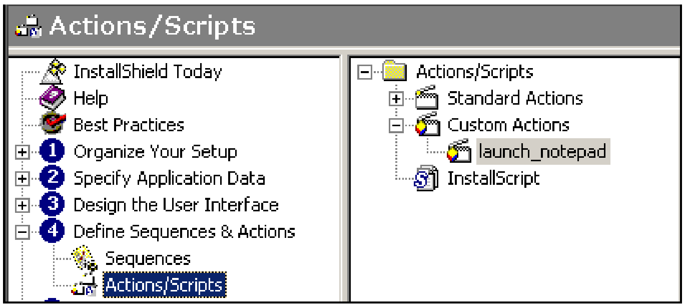
Scenario:

Let us suppose that your installation had the requirement to display a .txt file of release notes to the end user but only if they selected the 'Modify' option off the MaintenanceType dialog.

The first stage is to create a Custom Action *(off section ❹ 'Define Sequences & Actions')* that will launch Windows Notepad with the .txt file. For the purpose of this scenario we can use the 'Custom Action Wizard', shown below:



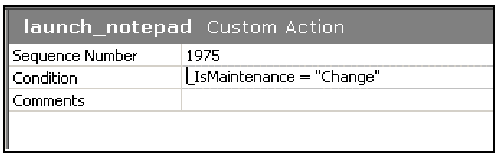
Once we have run through the 'Custom Action Wizard' we will have a newly created Custom Action, in this case called 'launch_notepad':



We now need to inse rt this Custom Action into the ‘Sequences’ view of our installation) *for example, into the Installation ->Execute Sequence).* Once inserted, we can then use the ‘_IsMaintenance’ Property as part of a Condition, which will control when the Custom Action will execute.

For our example, we recall that the scenario was to ensure that Windows Notepad was launched when the end-user selected the ‘Modify’ option.

Using the table on p8 we would see that the corresponding value for ‘_IsMaintenace’ is ‘Change’. Therefore, we write the following Condition and attach this to the Custom Action, as shown below:



launch_notepad Custom Action

Sequence Number	1975
Condition	_IsMaintenance = "Change"
Comments	

:

As a result of this Condition, the Custom Action ‘launch_notepad’ will only execute if the end-user selected the ‘Modify’ option off the ‘Program Maintenance’ dialog (known as ‘MaintenanceType’ within Windows Installer.