# InstallShield Tip: Installation and Uninstallation Conditions

Robert Dickau
Senior Techincal Trainer
Macrovision
www.macrovision.com/services/education

**Abstract**

Basic MSI projects created with InstallShield are database-driven, and not script-driven. One feature of MSI databases is that there are sequence tables that define the order of actions and dialog boxes to be performed and displayed. This article describes how to define Windows Installer conditions that enable you to specify that an action should be performed only during a first-time installation, a complete uninstallation, or only when a specific feature is being installed.

## Installation and Uninstallation

Instead of using an explicit script, Basic MSI setups determine the actions to perform and the order in which they will occur using sequence tables. In particular, the InstallUISequence and InstallExecuteSequence tables—the "User Interface" and "Execute" sequences—contain ordered lists of actions to perform during a normal installation (that is, an installation started by double-clicking an MSI database icon or by running the command **msiexec /i ProductName.msi**).

When working with the MSI sequence tables, an important idea is that the same sequences are used for first-time installations and maintenance installations (including uninstallation); there is no separate "uninstallation sequence". Therefore, any custom actions you schedule in the Installation sequences will by default run for both installation and uninstallation. The problem, of course, is that this behavior is undesirable for many custom actions, especially those involving launching or manipulating files installed by your installation.

To ensure an action runs only for a first-time installation, you can use the condition **Not Installed**. (Keep in mind that property names are case-sensitive. A common error is to mis-capitalize the property name as "Not INSTALLED"; because the all-uppercase property INSTALLED is typically undefined, the condition "Not INSTALLED" will always succeed.) The condition **Not Installed** is appropriate, for example, for a custom action that launches a Readme file or the application being installed.

The Installed property is set if the current product is already installed, either as a per-machine installation or a per-user installation for the current user. (Note that Windows Installer also defines a ProductState property, which additionally enables you to detect if the current product is installed for another user.) If you have a custom action that should run for any maintenance operation—modify, repair, or remove, but not a first-time installation—you can use the condition **Installed**.

©2002-2007 Macrovision Corporation. All rights reserved.

To detect a complete uninstallation, you can use the REMOVE property in a condition. The REMOVE property is set to a comma-separated list of features that are to be removed. During a complete uninstallation, REMOVE is set to the special string value ALL, and therefore you can use the condition **REMOVE="ALL"** to detect a complete uninstallation. (Note that this condition is valid only after the InstallValidate action in the InstallExecuteSequence table.)

Another option for detecting a complete uninstallation in the User Interface sequence is to use the _IsMaintenance property. The standard MaintenanceType dialog box offers Modify, Repair, and Remove radio buttons; the user's radio-button selection is stored in the _IsMaintenance property, with one of the values "Change", "Reinstall", or "Remove". In the User Interface sequence, then, you can give an action the condition **_IsMaintenance="Remove"**, anywhere after the MaintenanceType dialog box. (Because _IsMaintenance is a private property—its name contains lowercase letters—its value will be reset when execution switches from the User Interface sequence to the Execute sequence, and therefore it cannot be used in the Execute sequence to detect the installation type.)

To summarize, you can use the following conditions to detect different installation modes:

- First-time installation: **Not Installed**
- Any maintenance type: **Installed**
- Uninstallation: **REMOVE="ALL"** (after InstallValidate)

**Feature and Component Conditions**

In addition to detecting different types of installation for the entire product, it can sometimes be useful to detect if a specific feature or component is being installed or removed. To support this, Windows Installer provides special syntax in MSI database fields that use the Condition data type, such as the Condition field of a record in any of the sequence tables.

The most commonly used type of feature condition is the "feature action" type, in which the expression **&FeatureName** is set to a numeric value indicating the operation to be performed on the feature called "FeatureName". The possible values for &FeatureName are the following (valid after the CostFinalize action):

| | |
|---|---|
| -1 | No action (feature unchanged) |
| 1 | Feature advertised |
| 2 | Feature not installed (uninstalled) |
| 3 | Feature installed locally |
| 4 | Feature installed to run from source |

For example, to detect if the feature called "FeatureA" is selected to be installed locally, and that it wasn't already installed locally, you can use the condition **&FeatureA=3**. You can use this condition on an action in the sequence tables, or on a NewDialog control event on the Next button of (for example) the CustomSetup or SetupType dialog box.

Feature-action conditions enable you to determine if a feature's installation state is changing (in the example above, from "not installed" to "installed locally"). If you want to

©2002-2007 Macrovision Corporation. All rights reserved.

determine a feature's state, regardless of whether the feature's state is changing, you can use "feature state" conditions. Feature-state conditions use the form **!FeatureName=*n***, where *n* is one of the values from the table above.

Similarly, Windows Installer defines component-action and component-state conditional syntax, respectively **$ComponentName=*n*** and **?ComponentName=*n***, where again *n* is one of the values from the table above.

**Additional Considerations**

Some additional considerations when working with installation and uninstallation are the following:

- If the user performs a silent or limited-UI installation, Windows Installer carries out only the actions in the Execute sequence. Therefore, any actions placed in the User Interface sequence will be skipped, irrespective of any condition attached to them.
- Related to the previous note, if the user launches an uninstallation by clicking the Remove button on the Add/Remove Programs panel (in Windows 2000 or later), only the actions in the Execute sequence will be performed.
- Custom actions that launch documents or executables during a first-time installation are sometimes scheduled in the Execute sequence; however, a document's or executable's user interface is generally inappropriate for a silent installation. To ensure an action runs only during a full-UI installation, you can combine the conditions described in this article with a condition using the UILevel property. The UILevel property is set to 5 for a full-UI installation (with lower numeric values for silent, basic-UI, and reduced-UI installations), and therefore you can use a condition such as **(Not Installed) And (UILevel=5)** to detect a first-time installation running with a full user interface.

For more information, see the MSI Help Library page "Conditional Statement Syntax".

©2002-2007 Macrovision Corporation. All rights reserved.