InstallShield Tip: Saving MSI property values to make them available after the initial installation

Robert Dickau
Senior Technical Trainer
Flexera Software

Abstract

One aspect of Windows Installer's behavior that is occasionally surprising to developers is that the values of MSI properties set during installation are generally not available during maintenance mode or uninstallation. This article describes different approaches to saving MSI property values to make them available after the initial installation.

Property Basics

The initial values of Windows Installer properties can be stored in the Property and Directory tables of an MSI database (examples are ProductName, ProductVersion, INSTALLDIR, and the Add/Remove Programs properties having names beginning with ARP), or are initialized by the Windows Installer engine during setup initialization (examples are VersionNT, Version9X, ProgramFilesFolder, AdminUser, and Privileged).

At run time, properties can be created or changed in many different ways, including:

- At the msiexec.exe command line (as in `msiexec /i product.msi PROPERTYNAME="Value"`).
- By a user-interface control or control event (common examples are USERNAME, COMPANYNAME, and ALLUSERS).
- With a custom action, using `MsiSetProperty` or `Session.Property`.

At run time, Windows Installer passes only the values of public properties (those with all-uppercase names) from the User Interface sequence to the Execute sequence; and when an installation is complete, property values are generally lost.

Properties Automatically Saved after Installation

A handful of property values are automatically written to the target system during installation, and are available later using Windows Installer API functions or Automation properties. For example, the values of USERNAME, COMPANYNAME, and (if used) ProductID set during the initial installation are written to the target system's registry by the standard action RegisterUser. During a maintenance operation, you can read the values originally set by calling the **MsiGetUserInfo** API (from InstallScript or a C DLL).

Moreover, the values of ProductName, ProductVersion, Manufacturer, and most ARP properties are written to the target system during installation, and are available during a maintenance operation using the **MsiGetProductInfo** API function, or the **ProductInfo**

Automation property.

The value of the predefined property **ARPINSTALLLOCATION**, which is intended to represent the main installation directory for a product, is automatically written to the target system, and available later using MsiGetProductInfo or ProductInfo. However, by default this property does not have a value. To write the value of INSTALLDIR to the registry so that it can be read during a maintenance operation, you can create a set-a-property custom action that sets **ARPINSTALLLOCATION** to **[INSTALLDIR]**, adding the custom action to the Execute sequence after CostFinalize, with condition **Not Installed**. After doing so, you can read the value of INSTALLDIR during a maintenance operation using a VBScript custom action (for example) as follows:

```
str = Installer.ProductInfo(Property("ProductCode"), "InstallLocation")
```

In InstallScript, a similar call might appear as follows (assuming the variables `nBuffer` and `svINSTALLDIR` have been declared):

```
nBuffer = MAX_PATH + 1;
MsiGetProductInfo("{ProductCode}", INSTALLPROPERTY_INSTALLLOCATION,
svINSTALLDIR, nBuffer);
```

Developer 8.01 and later automatically include a custom action to set ARPINSTALLLOCATION.

Explicitly Saving and Reading Property Values

For properties not included in the small collection listed above, you will need to explicitly save the property value to the target system during installation, and read the value back during maintenance operations. Recall that you can use the condition **Installed** to ensure that a custom action runs only for a maintenance operation, and not for a first-time installation.

Common practice is to store desired property values in the registry. Because the Value field of the Registry table uses the Formatted data type, you can enter expressions such as [PROPERTYNAME] in the IDE Registry views, and the value of PROPERTYNAME will be written to the specified registry key. (For per-machine registry data, it is recommended values be stored in the application-information key **HKLM\Software\[Manufacturer]\[ProductName]\[ProductVersion]**.)

During a maintenance operation, you would then read the data back from the registry. The AppSearch and RegLocator tables enable you to read data from the registry, as do the `RegistryValue` method of the MSI Automation interface and the InstallScript function `RegDBGetKeyValueEx`.

To use the AppSearch table to read the registry, open the System Search view, right-click in the System Search table, and select Add. In the System Search Wizard, select the search type **Registry entry that contains other data**, specifying the root key, subkey, and value name in which you wrote the property value, and finally a property name in which to store the retrieved data.

Using InstallScript, the following code will read data from the registry (assuming the variables used have been declared):


```
// set the root key
RegDBSetDefaultRoot(HKEY_LOCAL_MACHINE);

// read the desired value data
RegDBGetKeyValueEx(
  "SOFTWARE\\SampleCo\\SampleApp\\1.0.0", // [in] subkey
  "SavedValue",                          // [in] value name

  nvType,                               // [out] value type
  svSavedValue,                         // [out] value, in string form
  nvSize);                              // [out] value size
```

After calling `RegDBGetKeyValueEx` as above, the output string variable `svSavedValue` will contain the value read from the registry. If desired, you can then call `MsiSetProperty` to assign the value you read to an MSI property.