Using InstallAware 7

To Publish Web Updates

November 2007

InstallAware

TABLE OF CONTENTS

Overview

This whitepaper describes how to use InstallAware for publishing software updates on the web. The software lifecycle has evolved to the point where updates are still expected to be delivered after a product's original release to manufacturing. To accommodate this requirement, developers are often tasked with building web update clients from scratch. These home-brewed web update clients consume significant development resources to build and maintain over time. InstallAware provides out-of-the-box web update capability to address this requirement at absolutely no effort or cost to you; whilst maintaining full customizability and extensibility in the good InstallAware tradition.

Unlike other web update mechanisms, InstallAware does not install an update service which constantly consumes system resources and memory, and increases the attack surface on the client machine. InstallAware does not force its branding or impose other limitations on the web update user experience either. While some web update mechanisms can be customized only through HTML templates, InstallAware lets you take full control over the entire update process – including not only the update user interface, but also the workflow. Updates may be shipped selectively to a subset of your user base, decided according to any custom set of parameters you may have.

InstallAware's out-of-the-box web update functionality is modeled after Windows Update to ensure optimal user experience and confidence. The web update client has been implemented entirely in InstallAware, and not a separate third party programming environment; which is a testament to the capabilities of InstallAware's MSIcode scripting and dialog design capabilities. This also makes the entire update process fully customizable and transparent. Since the full source codes for the update client are provided, it is very easy to inject custom behavior, or completely overhaul the update mechanism as necessary; whilst having pre-built code-blocks and update dialogs on hand to avoid costly re-engineering of common web update behavior.

Additionally, since the web update functionality is compiled into the main setup file, a separate update client does not need to be installed on target systems, saving space both on end-user machines and inside the setup file thanks to reduced payload.

Finally, just like Windows Update, InstallAware schedules web updates to be run on a date and time chosen by the end-user. The update client never consumes memory or other system resources in the background, since it runs only at this chosen time. The update download and installation process is very un-intrusive (again, just like Windows Update), with message balloons in the system tray providing helpful indicators as to the availability of updates and the progress of the download/installation. Again, all these out-of-the-box aspects of the update client may be easily customized simply by editing the existing MSIcode script and dialogs.

These advantages make InstallAware web updates the ideal technology to push updates to end-user systems after your product has been released to manufacturing, delivering you maximum cost savings without losing the ability to customize the web update process for your business processes and customer requirements.

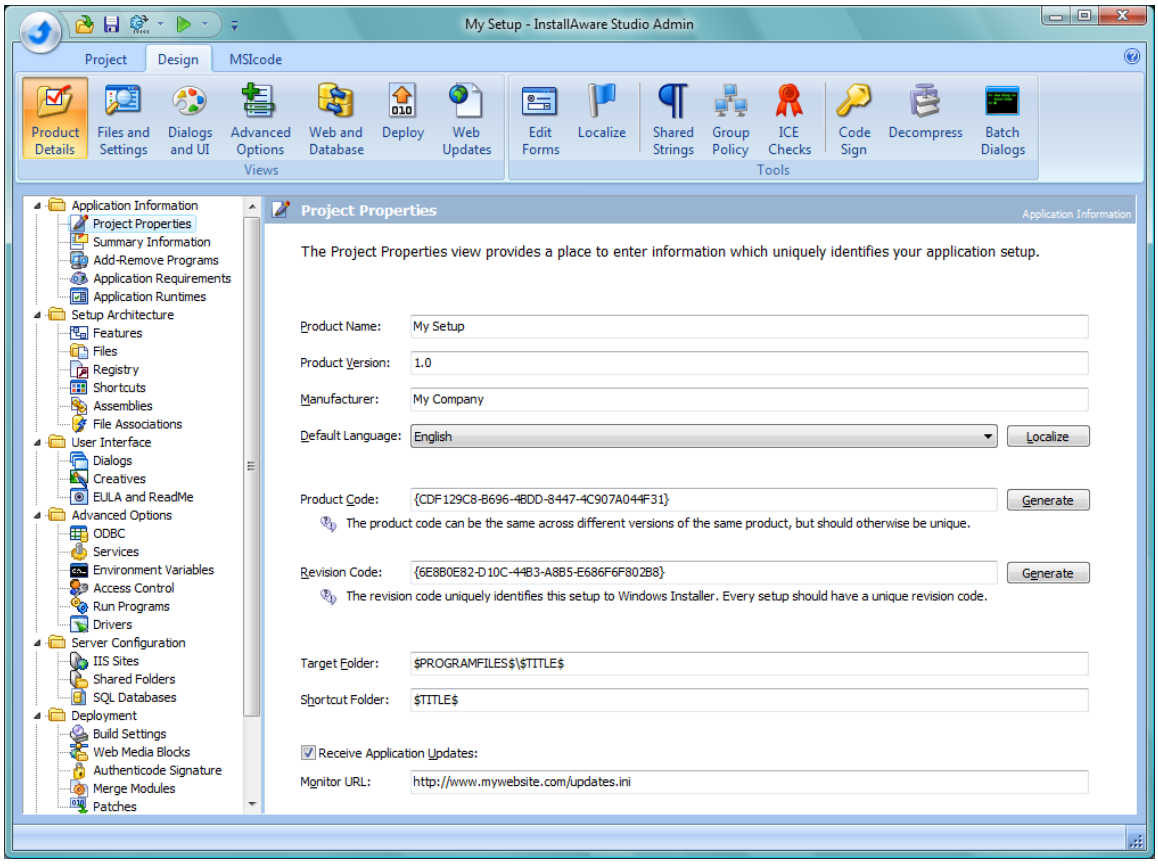
Enabling the Web Update Client

To enable web updates for products you are installing with InstallAware, perform the following steps.

1. Launch the InstallAware IDE using the Windows **Start Menu**, and open the setup project for the product that you wish to enable the web update client for.

2. On the **Design** tab, in the **Views** group, click the **Product Details** button. Choose the **Project Properties** design view under the **Application Information** heading.

3. Check the **Receive Application Updates** check-box. InstallAware suggests a starting URL for the **Monitor URL** field automatically. Revise the URL as necessary to point to your web server. Don't worry about the `updates.ini` file for now.



That's it! The InstallAware web update client is now enabled for your product.

Kinds of Updates

This document does not describe the creation of product updates themselves. There are no pre-set limitations on what a product update might be comprised of. An update might be a full `setup.exe` file that contains the full new version of your product; it might be a patch which upgrades older installed product versions to newer versions; or any other custom executable file which updates your product to the desired state when run:

- **Full Product Installers:** A full rebuild of your existing setup project can be served as an update to your end-users. When run, setup will automatically uninstall the old product version, treating this step as an installation prerequisite, and then move on to reinstall your application's latest version. Of course, this approach carries the drawback that your existing end-users have to download a full version installer, when they already have some bits on their system within their existing product installation.
- **Binary Patches:** If you would like to serve incremental binary patches instead, which contain only the changed data between your product versions, please refer to the InstallAware whitepaper "Using InstallAware to Patch Applications", which describes how to create patches for your products using InstallAware.
- **Custom Programs:** Alternately, an update may be a completely custom program file that carries out any sequence of actions as required by your update scenario. It does not have to be a setup or a patch, it does not have to be an InstallAware built file; nor does it need to update any files, registry keys, and so on. You have complete freedom in running any executable program you wish, as long as it performs the update operation you desire.

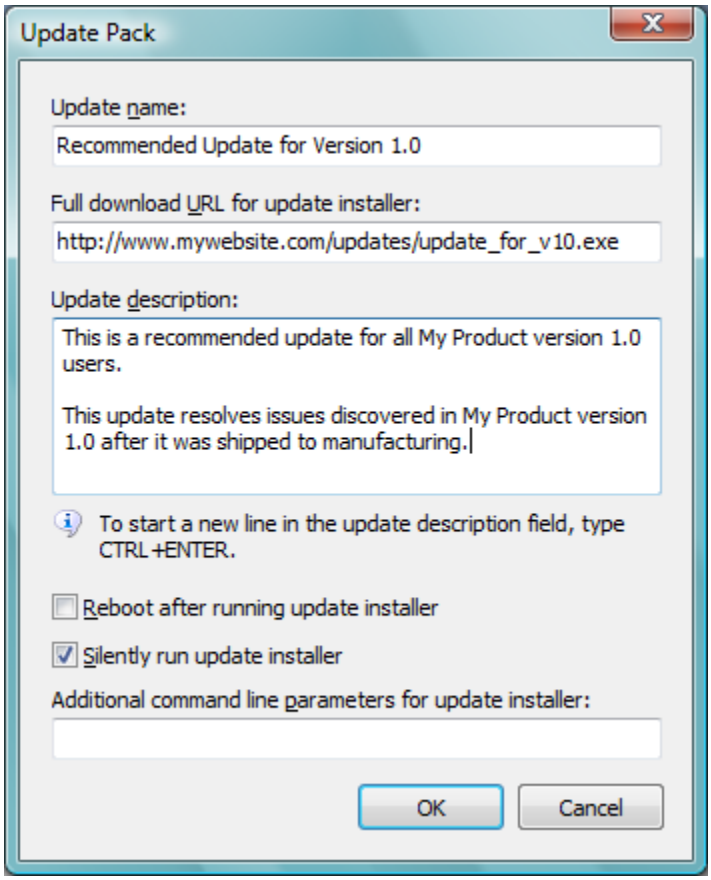
Continue to the next section once you have a couple of updates available for your application. For the purposes of completing this guide, you may simply rebuild your existing setup project a few times without changes, just to have several `setup.exe` files that serve as mock product updates. Before each rebuild, just make sure to increment the **Product Version** field in your project, so InstallAware can keep track of the changed "new version" numbers.

Update Packs

Each available product update is treated atomically in InstallAware and referred to as an "Update Pack". Follow these steps to define update packs for your product:

1. On the **Design** tab, in the **Views** group, click the **Web Updates** button. Choose the **Update Packs** design view under the **Web Updates** heading.

2. Click the **New** button to begin authoring a new Update Pack definition. The Update Pack dialog box appears.



In the **Update name** field, enter a descriptive name for this update. Your update will be displayed to end-users using this name when they check for updates.

In the **Full download URL for update installer** field, enter the web (or network) address where the actual update installer can be downloaded from at runtime. You may choose FTP, HTTP, or HTTPS sites for hosting your update files. Be sure to upload the update installer to your server using the exact filename that is specified in this field.

In the **Update description** field, enter the multi-line description for this update.

Check the **Reboot after running update installer** check-box if you'd like to restart the system after update installation. Just like Windows Update, the end-user will be prompted for a reboot at the end after all chosen updates have finished installing.

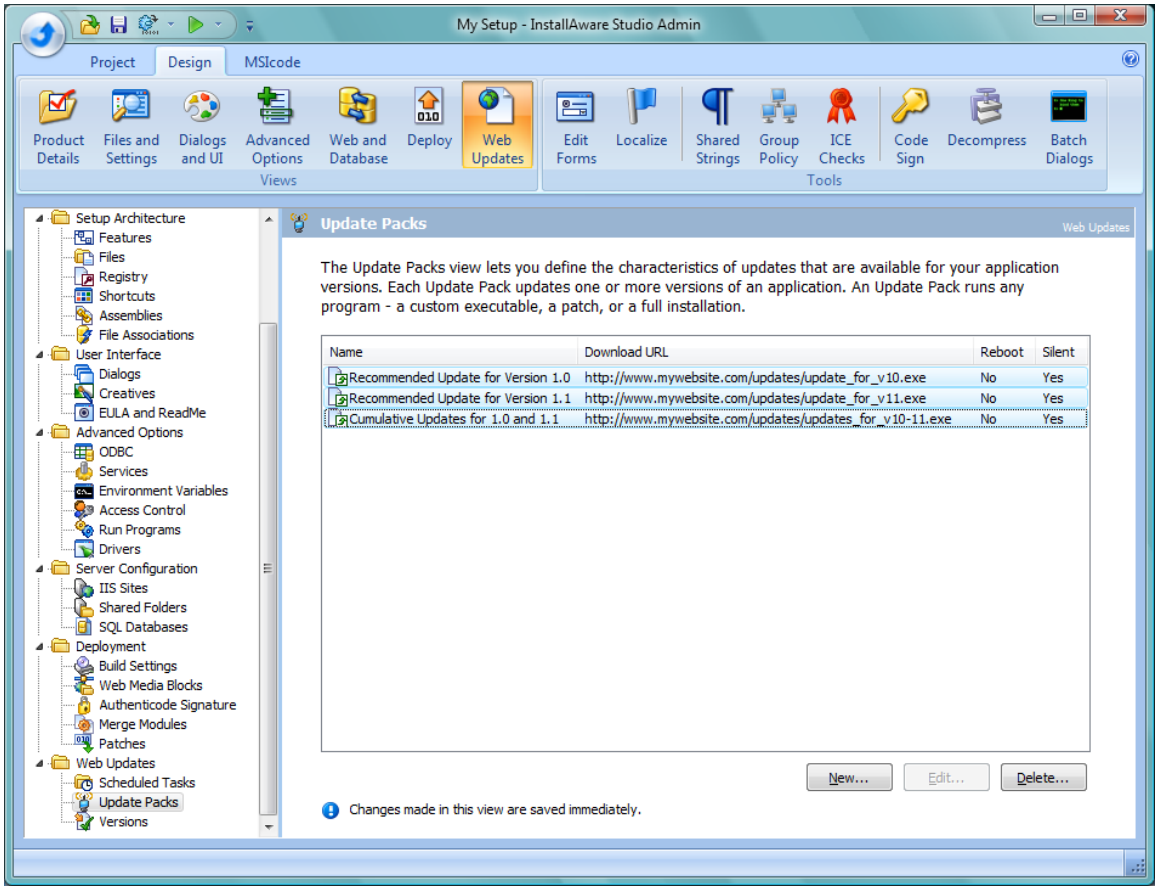
Check the **Silently run update installer** check-box if you'd like to run the update installer silently. InstallAware passes a `/s` command line switch to your update installer in order to facilitate silent installation, which is the standard command line parameter for silent InstallAware setups. If you are running a custom program, note that the `/s` switch may have no effect. Updates are generally installed silently and it is recommended for the optimal end-user experience that you run your updates silently as well.

If there are any other command line parameters you would like to pass to your update installer, enter them in the **Additional command line parameters for update installer** field.

Click **OK** when you are satisfied with your settings to save the Update Pack definition.

3. Each Update Pack may update one or more versions of your product as you see fit. An Update Pack does not have any built-in limitations as to the product version(s) it can be applied to, this happens at your own choosing.

   For instance, let's assume you have three versions of your product – versions 1.0, 1.1, and 1.2, with 1.2 being the latest shipping version. You could update version 1.0 users to version 1.1 with a patch, and then update version 1.1 users to version 1.2 with another patch. Alternately, you could update both version 1.0 and 1.1 users to version 1.2 with a cumulative update that upgrades both versions, without having to go from 1.0 to 1.1 first.



To test this idea, define two more update packs for your application. Call one of the packs `Recommended Update for Version 1.1` and the other `Cumulative Updates for 1.0 and 1.1`. Fill in the fields inside the **Update Pack** dialog accordingly for each update, so the **Update Packs** designer looks like above.

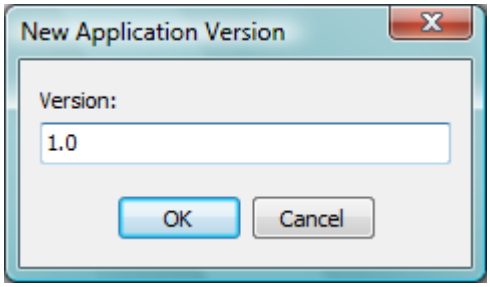
You have now finished describing the atomic updates that are available for your product.

Product Versions to Upgrade

The next step is to define the versions of your product that are already in use by your customer base and are the target of the updates. Follow these steps:

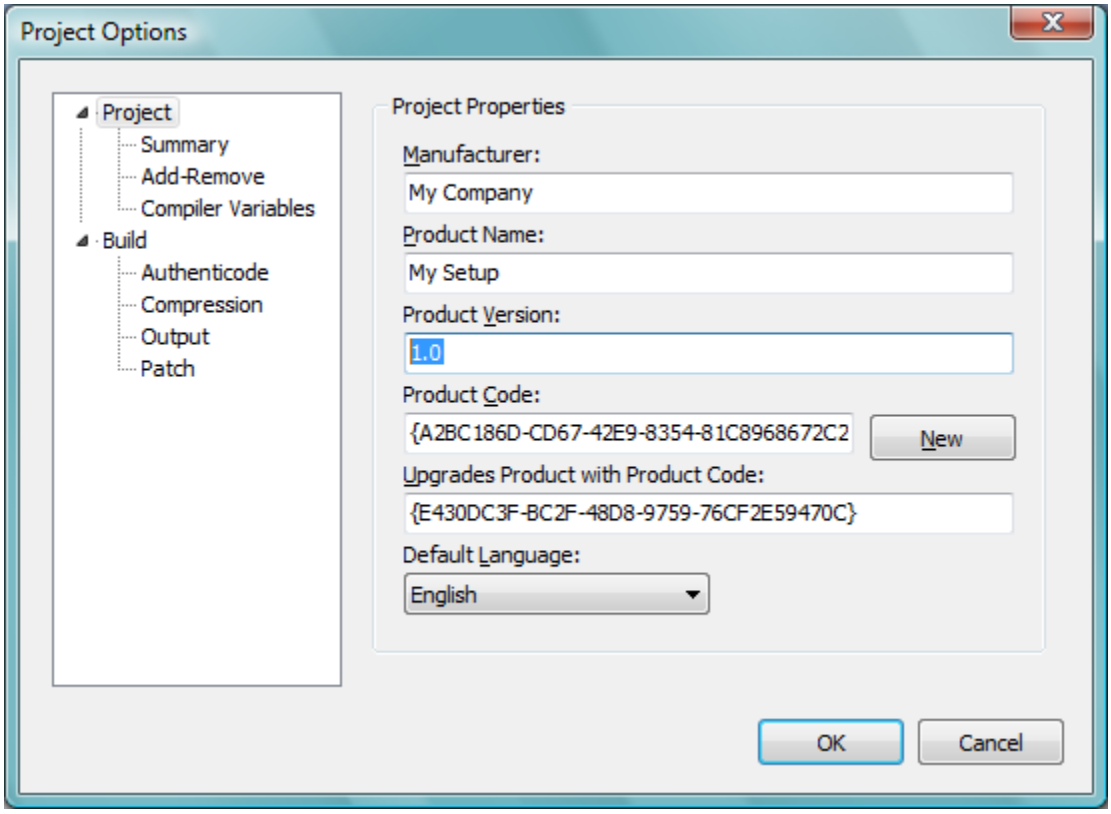
1. On the **Design** tab, in the **Views** group, click the **Web Updates** button. Choose the **Versions** design view under the **Web Updates** heading.

2. Click the **Add** button in the **Versions** designer to define a product version to upgrade. The **New Application Version** dialog appears.



One by one, enter the version number of each product to be upgraded, using this dialog. Continuing our previous example, use this dialog three times to define version numbers `1.0`, `1.1` and `1.2` for your product.

It's important to note that the version numbers you enter in this dialog must *exactly* match the version numbers you used for your product installers. To review the version number you are using in your current setup project, click the **InstallAware button**, and choose **Project Settings** to display the **Project Options** dialog. The **Project** page of this dialog displays the version number you are using, listed under the **Product Version** field:

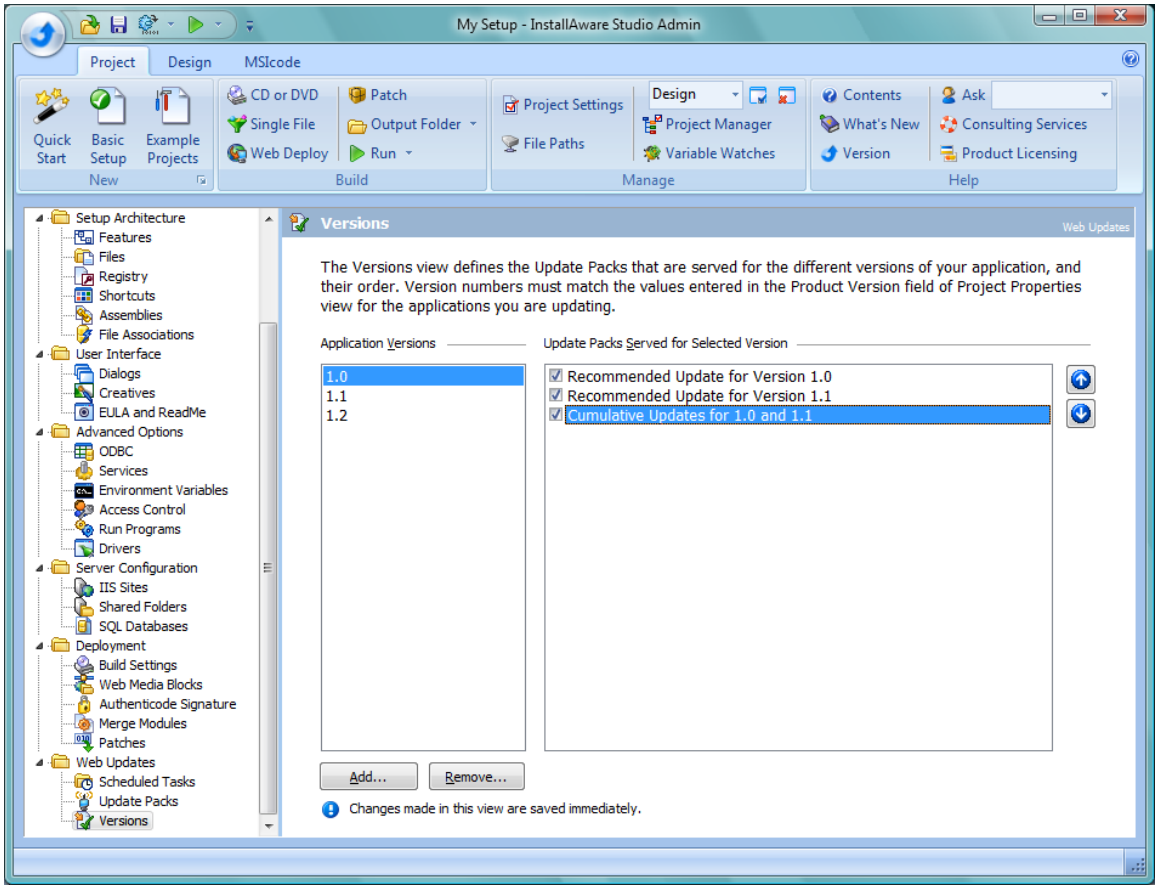


For instance, if the version of the product you wish to update was defined as `3.14.15926` in the **Project Options** dialog, use the exact same number in the **New Application Version** dialog. Using version numbers that do not precisely match will break the link between your updates and the products they are meant for.

Associating Update Packs with Product Versions

The last step is to link your updates with the products already deployed on the field. This is done through a point-and-click interface:

1. While still in the **Versions** designer, select a product version under the **Application Versions** heading. InstallAware lists all previously defined update packs to the right of your product version, under the **Update Packs Served for Selected Version** heading.



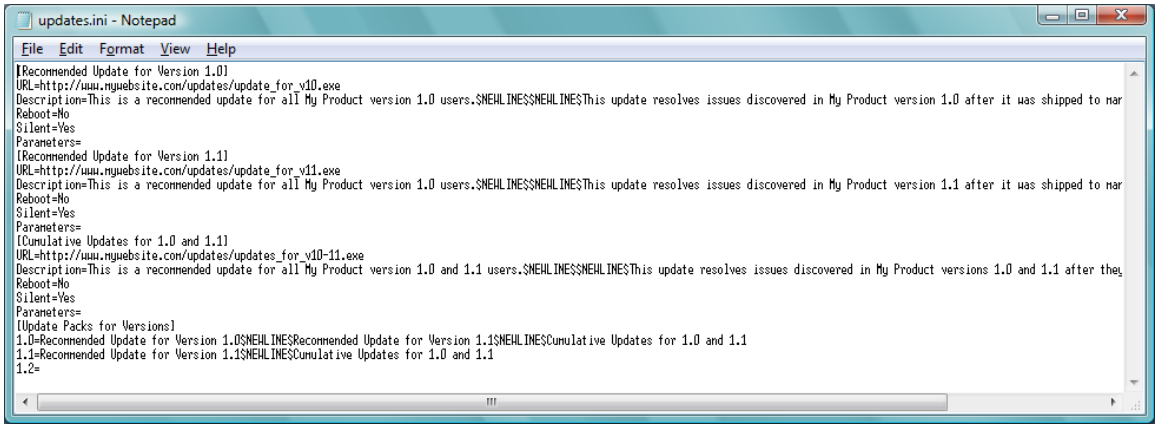
2. Check each update you wish to serve for the selected product version. You may check one or more updates for each version, depending on the updates that are applicable to that version. If no updates are available for that version (that is, the version you have selected is the most recent), simply make no selection, or clear any existing check-marks if there were previous selections.

   To change the order in which updates are installed, use the up/down arrow buttons found on the right of the update packs listing.

3. Repeat steps 1-2 above for each product version you have defined. This completes the association of your product versions with their applicable updates!

Going Live with Updates

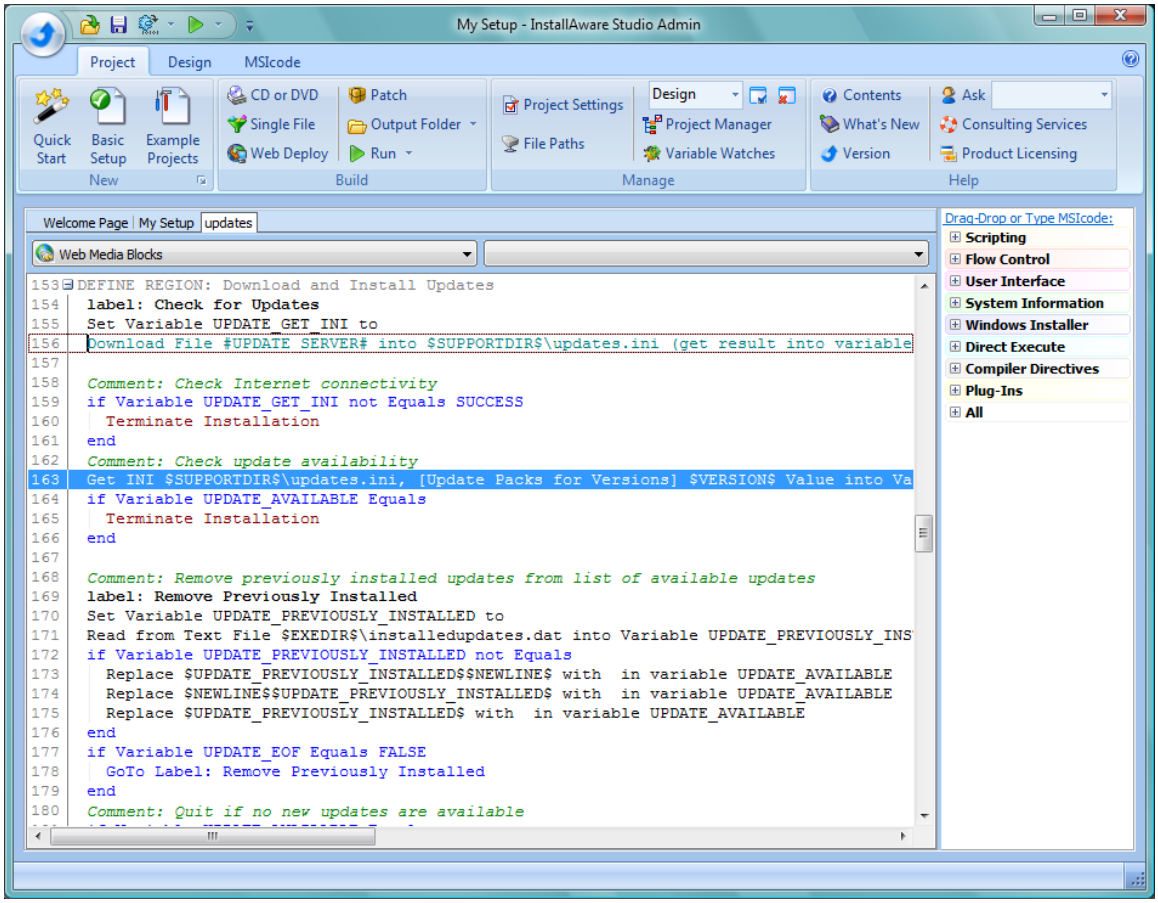
InstallAware records all of the work performed in the **Update Packs** and **Versions** design views of the IDE inside a file called `updates.ini`. This file is found inside your main project folder (it is also automatically copied into your build output folder every time you do a build). Browse to your project folder, and open up this file in Windows **Notepad** to see what it looks like:



As you can see, the `updates.ini` file is a plain-text file in the INI-file format describing each available update, faithful to the data visually entered using the InstallAware IDE. You may directly edit this file in Notepad and the InstallAware IDE will reflect your changes. You may also add custom fields to this file, especially in cases where you need to have more precise control over which updates are served to which of your end-users. Of course, the InstallAware IDE will not be able to visually represent your custom data, but it will preserve it.

The `updates.ini` file is completely independent from the remainder of your setup project; in fact it is not even included with your `setup.exe` file when you do a build. Any setup that has the web update client enabled will attempt to read this file from the URL specified in the **Monitor URL** field, found under the **Project Properties** design view of the IDE. You are free to rename this file into any custom name and extension. As long as the URL specified in the **Monitor URL** field exactly matches the location of your INI file, it will be downloaded and processed successfully.

Note that the **Monitor URL** field in the **Project Properties** designer actually sets the compiler variable `#UPDATE_SERVER#`, which the update client's MSIcode script uses at runtime to determine where to download the `updates.ini` file from.



Like all other compiler variables, this value can also be changed at the **Compiler Variables** page of the **Project Options** dialog.

Once you have uploaded the `updates.ini` file to its correct location, along with the update installers referred to by the Update Packs, your updates are live! The update clients for your installs on the field will automatically detect, download, and install your updates based upon the schedule and preferences of your end-users.

Deployment Scenarios

Manually Invoking the Update Client

As explained earlier, the update client is not separate from the actual `setup.exe` file that installs your product – but an embedded part of it. Setup determines if it's running in update client mode based on the command line switches passed to it, and branches out accordingly into either the update client script for the main setup script.

The default behavior of the update client, which you may freely customize by editing the MSIcode sources of the update script, is as follows:

1. After your application has been installed successfully, upon the next log-on of the user, a message balloon asks the end-user to pick a schedule for checking updates for your application.
2. The user double-clicks the message balloon icon in the system tray to display the update scheduling wizard. The user picks a schedule (daily/weekly) to check for updates, along with default preferences for automatically downloading/installing updates (or being prompted for each found update). The user also enters his/her logon credentials to enable the Windows scheduled task that InstallAware will create on the system for the update process.
3. InstallAware creates a scheduled task in Windows to run `setup.exe` in update client mode at the scheduled time under the specified user account. InstallAware also records the end-user's update download/installation preferences (automatic/manual).
4. At the scheduled time, Windows's task scheduler engine starts setup in update client mode. The update client silently checks for updates. If no updates are found, the update client silently quits. If updates are available, the update client downloads/installs the updates, automatically or with user approval based on the preferences set during the original update scheduling.
5. The update client does not run at all times in the background as a service. It only runs at the scheduled time, and remains in memory only while actually downloading/installing updates. This conserves system resources while also reducing the attack surface on the end-user system.

The update client may also be manually invoked, either from the command line or programmatically from your application, simply by running the `setup.exe` file with the correct command line parameters. The location of the update client (the `setup.exe` file) and the command line switches that trigger the various update client modes are explained below.

Please note that your application must already have been installed before updates can be scheduled or checked for. Do not invoke setup with the command line parameters described below before your application has been installed.

Determining the Location of the Update Client

For an **All Users** installation look under the following registry key:
`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\<Setup Project Title (value of the variable $TITLE$)>`

For a **Just Me** installation instead, look under the following registry key:
`HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\<Setup Project Title (value of the variable $TITLE$)>`

In both cases, read the data for the **UninstallString** string value. A typical value for this string is:
`"C:\Documents and Settings\<User Name>\Application Data\<Setup Revision GUID>\<Setup Project Title>.exe" REMOVE=TRUE MODIFY=FALSE`

Parse this string in your application to remove the command line parameters `REMOVE=TRUE MODIFY=FALSE` as well as the double quotes surrounding the main string. The final string that indicates the location of the update client (as well as your setup program) should be of this form:
`C:\Documents and Settings\<User Name>\Application Data\<Setup Revision GUID>\<Setup Project Title>.exe`

Programmatically Scheduling Updates

The command line switch to put `setup.exe` into update scheduling mode is `/updatesetup`. This command line switch may also be re-used after updates have already been scheduled to change the existing update schedule and preferences.

Note that the update scheduling process also determines what to do when updates are found – end-users choose between automatically downloading/installing available updates, or manually approving the download and installation of each individual update that is found. Even if you do not intend to offer updates for your end-users on a pre-set schedule, you should still run `setup.exe` at least once with the `/updatesetup` command line parameter to ensure that these important update installation preferences are initialized. Running the update client to download and install updates without this pre-initialization may result in runtime errors due to un-initialized update script variables, which are initialized during the update scheduling step.

You may force such a pre-initialization directly from your main setup script, right after your product installation completes, by following the steps below:

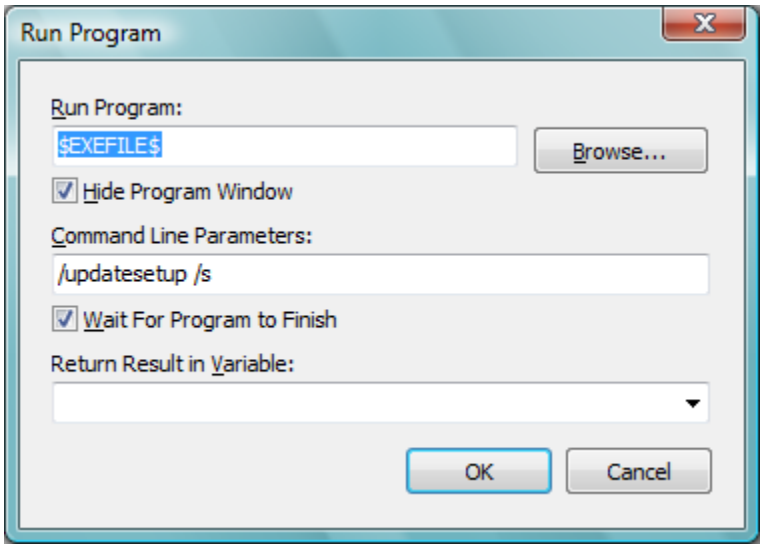
1. In the InstallAware IDE, switch to the **MSIcode** tab.

2. In the MSIcode editor, choose your main setup script, which is the page immediately to the right of the **Welcome Page**.

3. In the left code navigation drop-down, located directly above your MSIcode script, choose the `Comments` category.

4. In the right code navigation drop-down, choose the `TO-DO: Insert command that starts your application here` comment. This navigates the MSIcode editor to the part of your script that runs after a successful application installation (and never after a cancelled installation, or an uninstallation; which is not what we would want).

5. Drag-and-drop or type the **Run Program** MSIcode command into the highlighted region of the MSIcode script, and configure the Run Program dialog as in the screenshot below:



6. Click **OK**, which inserts this **Run Program** command into your MSIcode script. This new **Run Program** command makes use of the pre-defined `$EXEFILE$` variable to automatically resolve the location of the `setup.exe` file at runtime, and re-runs setup silently in update scheduling mode immediately after a successful installation of your application.

7. Save and rebuild your setup project to apply your changes.

This pre-initialization routine eliminates the need for your end-users to pick an update schedule during their next logon, and it also enables you to directly invoke `setup.exe` to check for updates at any time from your application.

Programmatically Checking for Updates

The command line switch to put `setup.exe` into update checking mode is `/update`. This command line switch may be used to check for updates programmatically from within your application. You may combine this switch with the `/s` silent installation command line parameter to automatically download and install updates that are found.

Ensure that `setup.exe` has been run at least once with the parameter `/updatesetup` before using this command line switch, as described above.

Returning Exit Codes from the Update Client

When you are invoking the update client from within your own application, and especially if you are running the update client silently, it may be desirable to obtain information on whether the client found any updates, successfully installed them, and so on. You may facilitate this by editing the update script, and using the MSIcode command **Terminate with Exit Code**, instead of the standard **Terminate Installation** command that the built-in update script uses to quit after finishing its run as the update client.

Simply search for the old command and replace it with the new command, along with your custom-defined exit codes:

1. In the InstallAware IDE, switch to the **MSIcode** tab.

2. Choose the `updates` script.

3. On the **MSIcode** tab, in the **Edit** group, click the **Find** button. Search for the text `Terminate Installation`.

4. Replace each found instance of the **Terminate Installation** command with the **Terminate with Exit Code** command using the MSIcode editor.

   For your convenience, below are some code snippets copied from the standard `updates` script, with comments inserted that highlight the meaning of each termination point:

```
if Variable WIZARD Equals CANCEL
  Terminate Installation
  // user cancelled update installation
else

…
  Terminate Installation
  // update scheduling successfully completed
END REGION Update Mode

if Variable UPDATE_GET_INI not Equals SUCCESS
  Terminate Installation
  // update server is offline
end

if Variable UPDATE_AVAILABLE Equals
  Terminate Installation
  // no updates found for product
end

if Variable ABORT Equals TRUE
  Terminate Installation
  // user aborted update installation
end

…
  Terminate Installation
  // updates successfully installed
END REGION Download and Install Updates
```

5. You may also want to replace the **Reboot Computer** command found once in the `updates` script with a **Terminate with Exit Code** command. Especially when running silently, you probably do not wish to restart the computer without the end-user's approval.

Note that there is no pre-determined rule as to what exit codes to use with the **Terminate with Exit Code** command. While `0` is used universally as the exit code indicating "success", all other exit codes have application dependent meanings. Simply define your own exit codes and check for them from your application invoking the update client. Example exit codes could be `1` for user abort, `2` for unavailable update server, `3` for no updates found, `4` for reboot required, and of course `0` for a successful operation. When defining your custom exit codes, keep in mind that Windows only permits numeric exit codes.

It is also important to note that the update client does not check for the exit codes returned by the actual update installer (the actual full setup, patch, or custom program applying your update on the system). You may want to return specific exit codes from the update installer itself, and check for those return codes right after **Run Program** finishes running the update installer in the MSIcode `updates` script.

Implementing Custom Update Filters

You may wish to serve updates only to qualified users – for instance, users who are on a subscription plan with your company. Alternately, you may have multiple editions of your product, and you may need to qualify the availability of updates not only based on the product version, but also on additional parameters, such as the language of the product. InstallAware makes it easy to add custom update filters to the standard update client, since you already have the full sources of the MSIcode update script.

In this section, we'll implement a language-based filter to illustrate how to fine-tune the delivery of your product updates using the pre-built InstallAware update client. Based on this example, you may add any further custom qualifiers to your update process, such as serial validation, subscription validation, and so on.

Extending Update Pack Definitions

Follow these steps to add a custom qualifier to the existing Update Pack definitions:

1. Open the `updates.ini` file created in the first part of this document using Windows **Notepad**. This INI file is found in your setup project folder.

2. Locate the INI file section created for the first update pack. This INI file section begins with the line `[Recommended Update for Version 1.0]`. Add a new INI file entry under this section: `Languages=English`.

3. Locate the INI file section created for the second update pack. This INI file section begins with the line `[Recommended Update for Version 1.1]`. Add a new INI file entry under this section: `Languages=German`.

4. Locate the INI file section created for the last update pack. This INI file section begins with the line `[Cumulative Updates for 1.0 and 1.1]`. Add a new INI file entry under this section: `Languages=English,German`.

5. Save the changes to your INI file and upload to your web server.

You have now added a new `Languages` qualifier to each Update Pack declaration. The update client can now be extended to access this information when reading and parsing the `updates.ini` file.

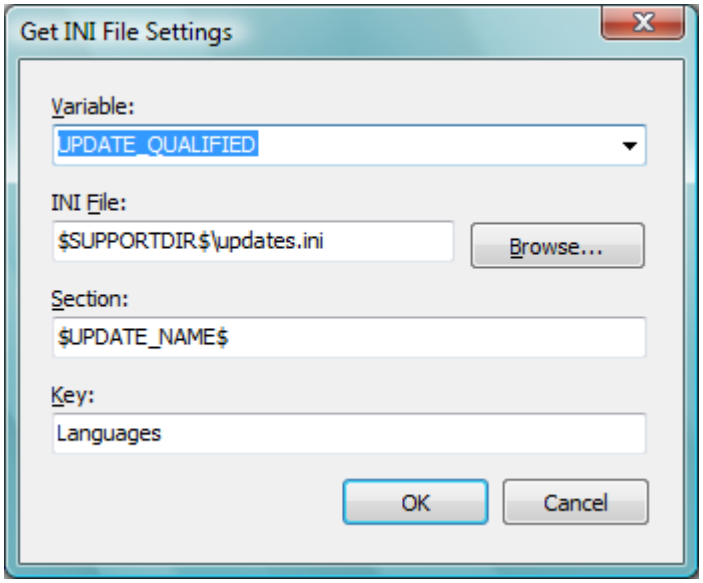
Extending the Update Client

To extend the update client, customize the MSIcode sources of the `updates` script, adding filtering logic into the pre-built script. To keep this document simple, the changes made below skip installing updates that aren't qualified, warning the end-user with a message box. A more advanced implementation can hide unqualified updates from the list of available updates altogether.

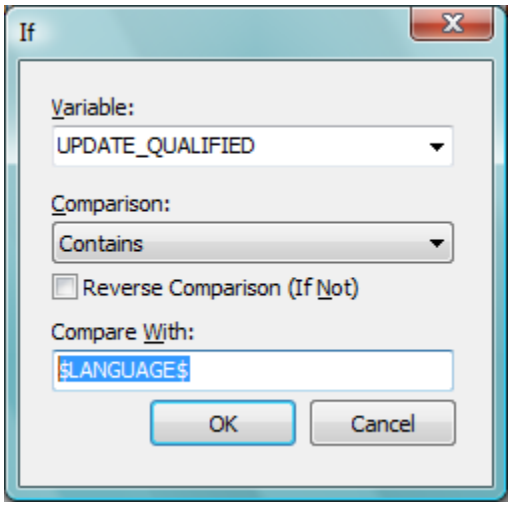
1. In the InstallAware IDE, switch to the **MSIcode** tab. Choose the `updates` script.

2. On the **MSIcode** tab, in the **Edit** group, click the **Find** button. Search for the text `Run Program $SUPPORTDIR$`. This is the part of the update script which runs the update installer.

3. Drag-and-drop or type the **Get INI File Settings** MSIcode command into the highlighted region of the MSIcode script, and configure the Get INI File Settings dialog as in the screenshot below:

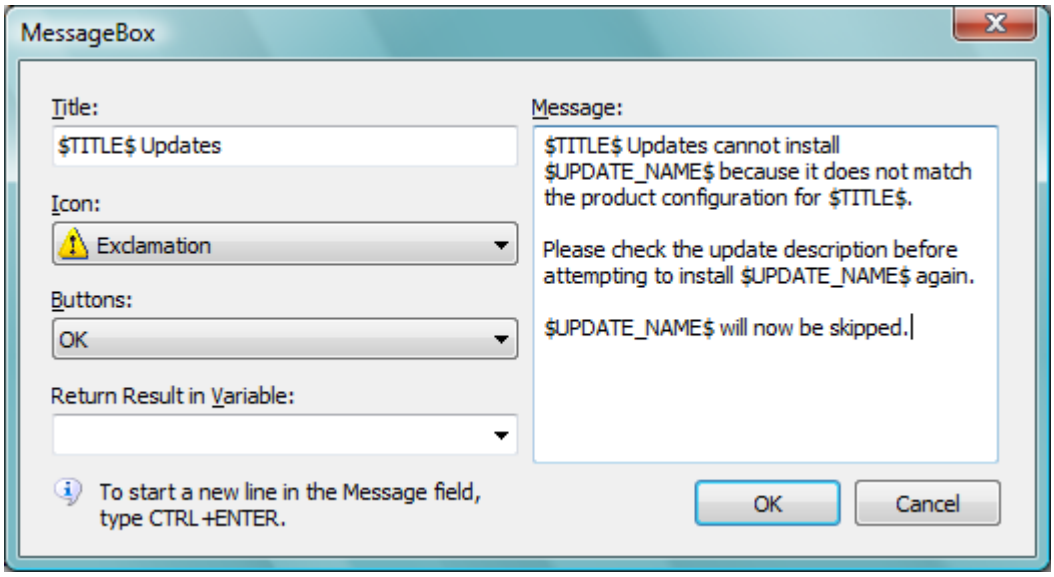


4. Now insert an **If** MSIcode command after the **Get INI File Settings** command, configured thus:



5. Insert an **Else** command right after the **Run Program** command.

6. Insert a **MessageBox** command after the **Else** command, configured as follows:



7. Insert an **End** command after the **MessageBox** command, closing the conditional clause started earlier with the **If** command.

8. Remember to save all your changes and rebuild your setup.

After completing the above steps, your MSIcode script should look as follows:

```
Get INI $SUPPORTDIR$\updates.ini, [$UPDATE_NAME$] Languages Value …
if Variable UPDATE_QUALIFIED Contains $LANGUAGE$
  Run Program $SUPPORTDIR$\$UPDATE_NAME$.exe …
else
  MessageBox: $TITLE$ Updates, $TITLE$ Updates cannot install …
end
```

In the above script, **Get INI File Settings** first reads from the INI file (which we updated in the previous section) the list of languages that are permitted for the update, and stores that list in the variable `$UPDATE_QUALIFIED$`. **If** then tests to see if the `$UPDATE_QUALIFIED$` variable contains the language that the pre-defined variable `$LANGUAGE$` evaluates to – this pre-defined variable always evaluates to the English name of the language setup is running under. If the evaluation is positive, the update is installed using **Run Program**, otherwise **MessageBox** notifies the end-user that this update is not applicable to their system.

For instance, for the `[Cumulative Updates for 1.0 and 1.1]` update pack, and an English language setup, the variable `$UPDATE_QUALIFIED$` will evaluate to the string literal `English,German`, and the variable `$LANGUAGE$` will evaluate to the string literal `English`. Therefore, since the larger string `English,German` contains the smaller string `English`, the update is applied.

This completes the process of implementing an additional filter for qualifying product updates. In just 5 lines of code, you customized the behavior of the pre-built update client for your custom requirements.

Additional Resources

Please visit the InstallAware website publications section at the following URL for more information on InstallAware technologies:

http://www.installaware.com/home/publications.asp

Whitepapers providing an in-depth analysis of InstallAware's scripting and web deployment technologies, as well as a Reviewer's Guide, are available at the above URL.

About InstallAware Software Corporation

Focusing solely on the Windows Installer (MSI) platform for Microsoft Windows operating systems, InstallAware is the premier provider of software installation tools for Internet deployment. Founded in 2003, InstallAware is supported by thousands of users worldwide, and is a Borland Integrated Partner. For more information, visit www.installaware.com.