Authoring Setup Packages

NOTE:

This documentation is a preliminary documentation. The primary purpose of this documentation is to enable the creation of setup packages for our products.

The 'Live Setup Team' is responsible for formulating setup guidelines for all the teams within Windows live. They will also provide certain setup code that we need to include in our setup files. **This documentation will get updated as and when we get more directives from the 'Live Setup Team'**.

***In the following documentation, all text shaded green are place holders for values that have to be entered by the user. In places marked by `ADD_GUID_HERE`, you have to provide a new unique GUID.**

How to create a MSI based setup package using WIX:

1.1 What is WIX:

The Windows Installer XML (WiX) is a toolset that builds Windows installation packages from XML source code. The toolset supports a command line environment that developers may integrate into their build processes to build MSI and MSM setup packages.

1.2 How to use the tool:

To use WIX you have author XML text files that end with the extension **.wxs**

Let us assume we author a file called **Sample.wxs**

At the CoreXT command prompt, execute

%EXTPATH%\wix\wixv3**candle** Sample.wxs

This will produce an intermediate file called **Sample.wixobj**. Now run the command,

%EXTPATH%\wix\wixv3**light** Sample.wixobj

This will produce a final setup package, **Sample.msi** (Sample.msi will be created in the current directory).

(**candle**.exe and **light**.exe are analogous to a compiler and linker.)

You can now run Sample.msi by double clicking on it. You can also run it from the command line by typing the command: **msiexec**. (Just type msiexec at the command prompt and that will tell you what parameters it takes as input).

2.1 How to author the source XML files (i.e. the .wxs files):

To simplify the process of authoring your .wxs files, copy and paste the following template into a text editor.

```
<?xml version='1.0' encoding='Windows-1252'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>

  <?define __PRODUCTNAME__          = ADD_NAME_HERE?>
  <?define __PRODUCTIDGUID__        = ADD_GUID_HERE ?>
  <?define __UPGRADEIDGUID__        = ADD_GUID_HERE ?>
  <?define __COMMENTS__             = ADD_COMMENTS_HERE?>
  <?define __PRODUCTDIRNAME__       = ADD_PRODUCT_SUBFOLDER_NAME_HERE?>

  <?define __INSTALLERVERSION__     = 300?>
  <?define __LIVEDIRNAME__          = "Windows Live"?>
  <?define PUBLISHEDSETUPDIR = $(env.inetroot)\published\setup ?>


  <Product
    Name='$(var.__PRODUCTNAME__)'
    Id='$(var.__PRODUCTIDGUID__)'
    UpgradeCode='$(var.__UPGRADEIDGUID__)'
    Language='1033'
    Codepage='1252'
    Version = '1.1.1.1'
    Manufacturer='Microsoft Co.'
    >

    <Package
      Keywords='Installer'
      Description='$(var.__PRODUCTNAME__) Setup Package'
      Comments='$(var.__COMMENTS__)'
      Manufacturer='Microsoft Co.'
      InstallerVersion='$(var.__INSTALLERVERSION__)'
      Languages='1033'
      Compressed='yes'
      SummaryCodepage='1252'
      />

<!-- Mandatory for all Windows Live teams: coresetup.wxi  -->
    <?include $(var.PUBLISHEDSETUPDIR)\inc\coresetup.wxi ?>

    <Media Id='1' Cabinet='Product.cab' EmbedCab='yes'/>

    <Directory Id='TARGETDIR' Name='SourceDir'>

      <Directory Id="ProgramMenuFolder" Name="Programs">
        <Directory Id="ProgramMenuSubFolder" Name="Windows Live" />
      </Directory>

      <Directory Id='ProgramFilesFolder' Name='ProgramFiles'>
        <Directory Id='LiveDir' Name='$(var.__LIVEDIRNAME__)'>
          <Directory Id='ProductDir' Name='$(var.__PRODUCTDIRNAME__)'>
            <Component Id='MainExecutable' Guid='ADD_GUID_HERE'>

                ADD_YOUR_FILES_REGKEYS_ETC_HERE


              <RemoveFolder Id='ProgramMenuSubFolderId'
Directory='ProgramMenuSubFolder' On='uninstall' />
              <RemoveFolder Id='ProductDirId' Directory='ProductDir'
```

```
On='uninstall' />

            </Component>
          </Directory>
        </Directory>
      </Directory>
    </Directory>



    <Feature Id='FullSetup' Title='$(var.__PRODUCTNAME__)' Level='1'>
      <ComponentRef Id='MainExecutable' />
    </Feature>

  </Product>
</Wix>
```

Figure. 1

NOTE: THIS TEMPLATE WILL UNDERGO UPDATES AS AND WHEN MORE DIRECTIVES/GUIDELINES ARE PROVIDED BY THE LIVE SETUP TEAM. IT IS EACH TEAM's RESPONSIBILITY TO MAKE SURE THAT THEIR WXS FILE MATHCHES WITH WHAT IS IN THIS TEMPLATE. YOU SHOULD PROVIDE VALUES FOR THE TEXT IN GREEN, AND YOU CAN ADD MORE COMPONENTS. HOWEVER, DO NOT MODIFY/DELETE THE TAGS THAT ARE DEFINED IN THIS TEMPLATE.

NOTE: `__PRODUCTIDGUID__` **and** `__UPGRADEIDGUID__` **should not undergo any changes for the lifetime of the product.**

This template will enable you to create one ***component*** that will be installed in the following folder:

Program Files -> Windows Live -> $(var. `__PRODUCTDIRNAME__`)

where the value of `__PRODUCTDIRNAME__` is supplied by you. This folder will be created for you during install, and will be deleted during uninstall.

You will have to add XML tags for the files, regkeys, etc that belong to your component in the section marked by `ADD_YOUR_FILES_REGKEYS_ETC_HERE`.
Sections **3.1, 3.2 and 3.3** will describe how to add files, shortcuts and regkeys.
For an example of a complete .wxs file, look at:

%_NTROOT%\client\personalMedia\shared\library\PhotoLibrary\PhotoLibrary.wxs

Note: The live setup team will implement tasks such as making sure there is enough disk space, the OS version is correct, the user has the right permissions,etc.

2.2 So what is a *Component* in WIX:

A ***Component*** is the atomic unit of things to be installed. It consists of resources—files, registry keys, shortcuts or anything else—***that should always be installed as a single unit***.

Each feature team will have to decide on how they plan to divide their binaries into components. The following links will help:

1) Organizing Applications into Components:

2) Defining Installer Components

3.1 Adding files:
For each file that belongs to your <Component/>, add a <File/> tag as follows:

```
<File Id='NAME_OF_FILE' Name='NAME_OF_FILE' DiskId='1'
Source='$(env._NTTREE)\$(env.bldPropSrc)\$(env.BUILDTYPE)\personalMedia
\NAME_OF_FILE' Vital='yes'/>
```

Where NAME_OF_FILE is the complete name of your file (including the extension).

The Source attribute needs to have the full path to the file. In the above example the file will be picked up from the folder specified in the _NTTREE environment variable in CoreXT (which in my case evaluates to F:\wlx\dmx.binaries). This might be different in your case (for example, if you have separate folders for retail and debug binaries). You have to ensure that you have provided the correct path to the file.

Add the attribute Vital='yes' only if the installation cannot proceed unless the file is successfully installed. If the Vital attribute is set to 'yes', the user will have no option to ignore an error installing this file. If an error occurs, they can merely retry to install the file or abort the installation.

For examples look at PhotoLibrary.wxs.
For additional attributes of the <File/> tag, please refer to the WIX documentation (link to the documentation is provided at the end of the document).

3.2 Adding Shortcuts:
First let us define the icon that we want to display as our shortcut. Generally, this icon will be a resource in your executable (this will be the icon used to display your executable in Windows Explorer). To use this icon, add the following line as a child of the `<Product/>` tag:

```
<Icon Id="ProductIcon.exe" SourceFile="$(env._NTTREE)\NAME_OF_FILE.exe" />
```

SourceFile is the full path to your icon file. Most probably this will be the full path to the executable, since the icon is mostly built in as a resource in the executable.
NOTE: The Icon identifier should have the same extension as the file that it points at. For example, a shortcut to an executable (e.g. "my.exe") should reference an Icon with identifier like "MyIcon.exe"

Now, before you add a shortcut for an executable, add the `KeyPath="yes"` attribute to the <File/> tag of that executable.
Eg. If you want a shortcut that points to `WLXPhotoGallery.exe` , add `KeyPath="yes"` as an attribute.

```
<File Id="WLXPhotoGallery.exe" Name="WLXPhotoGallery.exe" KeyPath="yes"
Source="$(env._NTTREE)\ WLXPhotoGallery.exe">
```

NOTE: The shortcuts must be defined inside the same <Component/> as the file they point too.

Now add a shortcut to your executable under "Start Menu -> Program Files -> Windows Live" as follows:

```
<Shortcut Id='StartMenuShortcut' Directory='ProgramMenuDir' Name='NAME_OF_PROGRAM'
Advertise='yes' Icon='ProductIcon.exe' />
```

3.3 Adding RegKeys:

Add a registry key as follows:

```
<RegistryKey Root=" ROOT_HIVE " Key=" FULL_KEY_PATH"
Action="createAndRemoveOnUninstall">
        <RegistryValue Value="DATA_VALUE1" Type="DATA_TYPE1" Action="write" />
        <RegistryValue Name="DATA_ITEM2" Value="DATA_VALUE2"
        Type="DATA_TYPE2" Action="write" />
</RegistryKey>
```

The above regkey would appear in a .reg file as follows:

```
[ROOT_HIVE\ FULL_KEY_PATH]
@=" DATA_TYPE1: DATA_VALUE1"
"DATA_ITEM2"=" DATA_TYPE2: DATA_VALUE2"
```

For example,

```
<RegistryKey Root="HKCR" Key="Software\Microsoft\Scrunch\Video"
Action="createAndRemoveOnUninstall">
        <RegistryValue Value="Windows Live" Type="string" />
</RegistryKey>
```

Will produce the key HKEY_CLASSES_ROOT\Software\Microsoft\Scrunch\Video and set the "(Default)" registry value under this key equal to "Windows Live".

If, however, you have a dll that does self-registration (i.e. implements DllRegisterServer), you can use the tool '**heat**' that comes with WIX.

Other non-COM related regkeys should go under (HKLM/HKCU)/Software/Microsoft/Windows Live/<ProductName>. Define a variable with this value in the main .wxs file, and use the variable in all other places where you define application regkeys.

Eg, `<?define __HKLMROOT__ = HKLM/Software/Microsoft/Windows Live/Bla?>`

3.4: How to use heat.exe:

Let's assume you have a dll MyBinary.dll that performs self-registration.
In the CoreXT command prompt, run the command

\%EXTPATH%\wix\wixv3\heat file MyBinary.dll –out Temp.wxs

This will produce the file Temp.wxs, which is a complete setup file. However, we cannot use it as is. Hence we will copy the required portion from that file, and paste it into our .wxs file.

Open Temp.wxs, and you should see the following inside:

```
.
.
.
<Component Id="MyBinary.dll" Guid="PUT-GUID-HERE">
```

```
            <File Id="MyBinary.dll" …
            .
            .
            .
            </File>
      </Component>
       .
       .
       .
```

You will see that your dll and its regkeys have been put inside their own <Component>. Cut and paste this <Component> (everything shaded in purple above) inside the <Directory> tags that define the directory where the dll should be installed.

NOTE: Please add a new guid for the place holder represented by ADD_GUID_HERE. Also, add the Vital='yes' attribute if required. Also, you will need to define the source attribute using environment variables i.e it should not be a hard coded value of the type "C:\.....".

4.1 Adding files to different folders and creating separate Components:

In the above example we added files to the same folder, and so we created only one component. If however, we want to add files to different folders, we will have to create separate components for each folder (*note: you might decide to have more than one component in the same folder. But you will need at least one component per folder*).

Each component has to be defined as a child of the <Directory Id='TARGETDIR' Name='SourceDir'/> tag.

For each component, begin by defining its directory structure using hierarchically nested <Directory/> tags. For our convenience, the installer environment provides predefined names for some of the common target locations (such as "Program Files", "System32", etc). You will use these predefined names in the Id attribute of the corresponding <Directory/> tag.

NOTE: In a <Directory/> tag, the value of the Name attribute is the name given to that folder. However, when you use a windows installer predefined name for the Id attribute of a Directory, the Name attribute becomes irrelevant. So just fill in some meaningful value.

For an example of how to add a second component, please look at the ThumbCacheComponent in PhotoLibrary.wxs.

For each component in your setup file, you need to add the corresponding <ComponentRef/> tag inside the <Feature /> tag.

E.g.,

```
      <Feature Id='Complete' Level='1'>
            <ComponentRef Id='Component1' />
            <ComponentRef Id='Component2' />
            .
            .
      </Feature>
```

5.1 Authoring WIX files for standalone binaries (i.e. binaries that do not constitute a complete application):

Assume you are the author of a dll that is part of some application that has its own setup file. However, you want to author and maintain the setup (and registration) portion of your dll. Then you can create a separate WIX file that will get ***included*** into the setup file of the application that uses your dll.

In this case, you will define a **.wxi** file. This file will then get included into the **.wxs** file defining the component.

Start with the following template for your **.wxi** file:

```
<Include>
            ADD_STUFF_HERE
</Include>
```

In the space denoted by `ADD_STUFF_HERE`, define the resources that belong to your binary/binaries as defined in sections 3.1 and 3.3 (section 3.2 will not apply to this case since you should not have any shortcuts for your binary).

Now inside the **.wxs** file, inside the <Component> tag of the Component that your file belongs to, include the **.wxi** file as follows:

```
<Component Id="MyComponent" …

        <?include FULL_PATH_TO_WXI_FILE\MyDll.wxi ?>          .
        .
        .
        .
</Component>
```

However, if you decide that your binaries (and its associated resources) need to be packaged as a separate <Component>, in the space denoted by `ADD_STUFF_HERE`, define your <Component> as described in section 4.1.
Now inside the main **.wxs** file, inside the `<Directory Id='TARGETDIR' Name='SourceDir'>` tag, include the **.wxi** file as follows:

```
<Directory Id='TARGETDIR' Name='SourceDir'>

        <?include FULL_PATH_TO_WXI_FILE\MyDll.wxi ?>          .
        .
        .
        .
</Directory>
```

NOTE: We cannot declare the same `<Directory>` at two places. To refer to a `<Directory>` defined in the main **.wxs** file, you will need to use the `<DirectoryRef/>` tag.

Example,

Consider the “Windows Live” folder defined in our main source file:

```
<Directory Id='LiveDir' Name='$(var.__LIVEDIRNAME__)'>
```

Now, if we were writing a .**wxi** file that was included in the main source file, we would install a component under some sub-folder under “Windows Live”, as follows:

```
<DirectoryRef Id="LiveDir">
    <Directory Id=…
        .
        .
        .
    </Directory>
</DirectoryRef>
```

The Id attribute of <DirectoryRef/> is the identifier of the Directory element to reference.

6.1 Incorporating Setup into the Build process :

Under the ***personalMedia*** folder in your enlistment, you should find a folder called ***Setup***. Under this folder create a subfolder for your product (you should find that there already exists a subfolder for PhotoLibrary).

Add your .wxs file (along with supporting .wxi files) into this subfolder.

In this subfolder you also have to define the following files:

1) Add a ***sources*** file with the following contents:

```
!INCLUDE $(INETROOT)\build\paths.all
!INCLUDE $(INETROOT)\build\sources.all

WIX_VERSION= 3.0.2526.0

TARGETNAME      = NAME_OF_YOUR_MSI
TARGETTYPE      = NOTARGET

PASS1_LINK      =
SOURCES         =
NTTARGETFILES   = $(TARGETNAME).wixobj $(TARGETNAME).msi

SYNCHRONIZE_DRAIN  = 1

WIX_REFERENCES  = $(INETROOT)\published\setup\lib\CoreSetup.wixlib
MISC_FILES      = *.msi
```

NAME_OF_YOUR_MSI should be the name of the msi file produced. For example, TARGETNAME = PhotoLibrary will produce an msi package called PhotoLibrary.msi.

You can also add candle and light flags in you sources file. For this and more info look at the CoreXT documentation (download corext.chm from http://stspine/corext/xt/default.aspx and search for “WIX Version 2”).

2) Add a ***makefile*** with the following contents:

```
!INCLUDE $(NTMAKEENV)\makefile.def
```

3) Add a ***makefile.inc*** file with the following contents:

```
!INCLUDE $(INETROOT)\build\makefile.inc
```

Finally, in the ***dirs*** file in ***the personalMedia\Setup*** folder, add your subdirectory.

7.1 Learn more about WIX:

If you know you will be doing a bunch of setup work, install the latest version of WIX from http://sourceforge.net/project/downloading.php?group_id=105970&use_mirror=umn&filename=Wix-3.0.2420.0.msi.zip&11548603

The help file that comes with the WIX package is very useful, especially to understand the WIX XML schema and to learn about the various XML attributes.

You can find an excellent tutorial at http://www.tramontana.co.hu/wix/

General info can be found at http://wix.sourceforge.net/

7.2 More Complicated Tasks:

Please contact anandi(or the WIX documentation) for tasks that are not covered in this document. Examples of some such tasks are:

- If your application handles its own file data type, you will need to register a file association for it.
- Checking if your dependencies are installed.
- Checking OS version since setup might be different for XP and Vista.
- Because folders created by the installer are deleted when they become empty, you must author a <CreateFolder/> entry inside your component to create an empty folder.

Also, the following wiki page has been created for sharing setup tips. Before authoring your file, refer to this wiki for style guidelines:

http://wiki/default.aspx/Microsoft.Projects.DigitalMemories/SetupUsingWIX.html

If this documentation has any errors, please send an email to santoshchapaneri (sachapan@microsoft.com)