

Tech Note 518

Managing Custom Script Function Libraries for Use in Application Server

All Tech Notes and KBCD documents and software are provided "as is" without warranty of any kind. See the [Terms of Use](#) for more information.

Topic#: 002252

Created: February 2008

Introduction

This *Tech Note* provides guidelines for developing and managing custom .NET Class Libraries, which are used as Application Server Script Function Libraries. This *Tech Note* is especially helpful when the .NET Class Library is modified and must be re-imported into the ArchestrA IDE. Steps are also provided to package the .aaSLIB file in case there are dependent files for the .NET Class Library.

Application Versions

- Industrial Application Server 2.1 P02
- Wonderware Application Server 3.0, 3.0 P01
- VisualBasic 6 SP5 for VB6 COM dll
- Visual C++ 6.0 SP5 for Simple Win32 dll
- Visual Studio2003, C# for Application Server Script Library

Scenario 1: Importing All Dependent Files

This scenario describes a common import operation that includes the following DLLs (Figure 1 below):

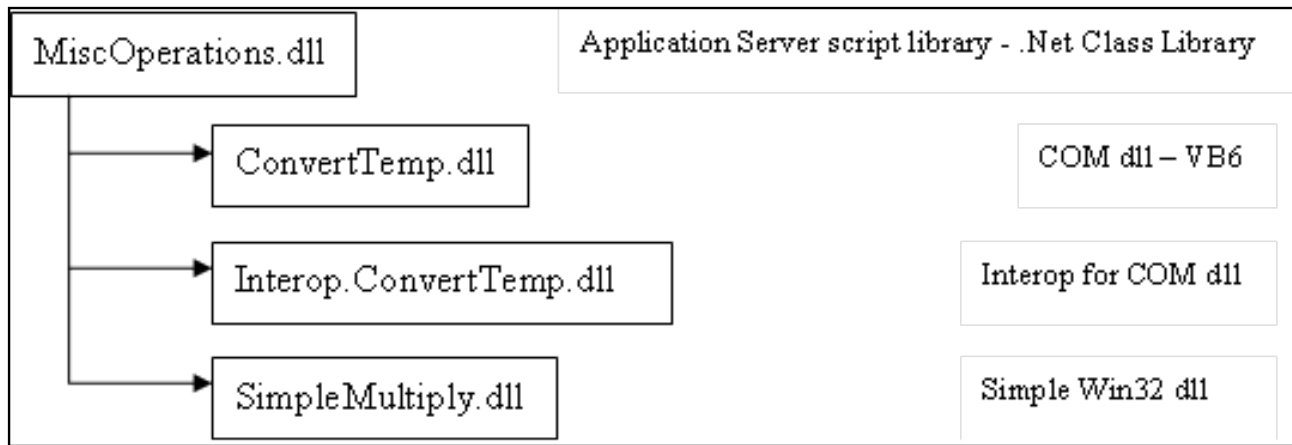


Figure 1: Scenario One Class Library List

- **MiscOperations.dll:** A .NET Class Library exposes methods that can be used as script functions in Application Server. It is an Application Server script library, a .NET Class Library developed in C# using VS2003.
- **ConvertTemp.dll & Interop.ConvertTemp.dll:** This .NET Class Library uses methods from a COM dll and interop file. The COM dll is developed in VB6.
- **SimpleMultiply.dll:** A dependent file which is a simple win32 dll with exported functions. This dll is developed in VS 6.0 C++.

When the **MiscOperations.dll** .NET Class Library is imported into the IDE, the import operation succeeds with the following results:

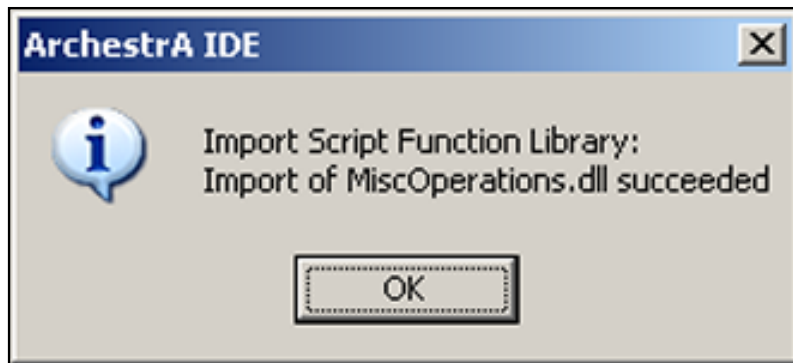


Figure 2: Import Successful

However, when a UserDefined Object (UDO) is created and deployed with a Script that uses a method from the .NET Class Library, the Script fails and generates the following error message in the SMC logger:

```
135378782 1/16/2008 2:48:23 PM 275 3020 3544 Error ScriptRuntime UserDefined_001.rr: Script performed an illegal operation.
```

135378783 1/16/2008 2:48:23 PM 275 3020 3544 Error ScriptRuntime UserDefined_001.rr:
MiscOperations: Unable to load DLL (SimpleMultiply).

The error indicates that the Script function used is dependent on another DLL called **SimpleMultiply.dll**.

Solution

To use all the methods from the .NET Class Library, create an **.aaSLIB** file to package all the dependent files. When this **.aaSLIB** file is then imported in ArchestraA IDE, all the dependent files are automatically copied at the relevant directory location.

To create an **.aaSLIB** file

1. Export the **MiscOperations** script function library. This operation generates a file called **MiscOperations.aaSLIB** in the designated directory.
2. Manually modify the XML file in an **.aaSLIB** file to add dependent files and also to designate a script function library as a COM object requiring registration:
 - o In Windows Explorer, rename the **MiscOperations.aaSLIB** file to **MiscOperations.aaSLIB.cab**.
 - o Create a working folder on your computer as shown below.

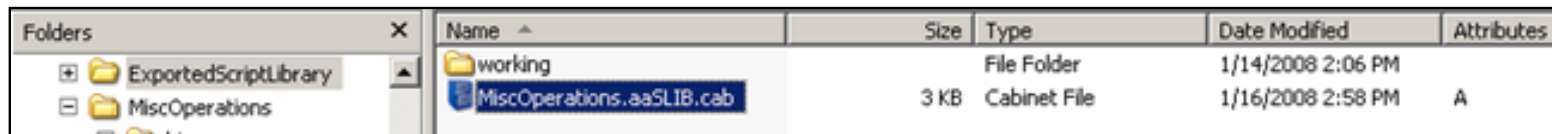


Figure 3: Working File

- o Double-click the **.cab** file. The files included in the **.cab** file appear.
- o Select all files, right-click and click **Extract**.
- o Browse to the **working** folder and click **Extract**.

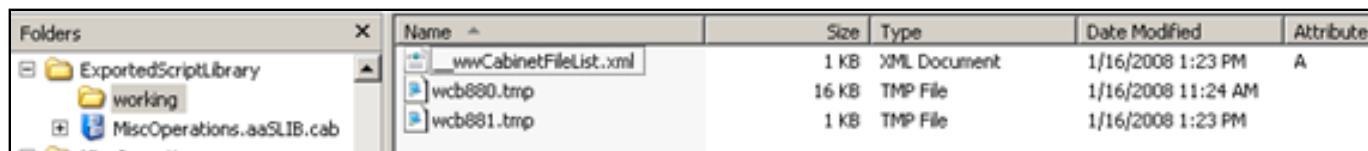


Figure 4: Extract .CAB File Contents

- o In Windows Explorer, browse to the **working** folder and open the **__wwCabinetFileList.xml** file using

Internet Explorer.

```
- <WWCabinetFileList>
  <FileItem FileName="MiscOperations.dll" FileCode="wcb880.tmp" />
  <FileItem FileName="MiscOperations.xml" FileCode="wcb881.tmp" />
</WWCabinetFileList>
```

Figure 5: Viewing the XML File Using Internet Explorer

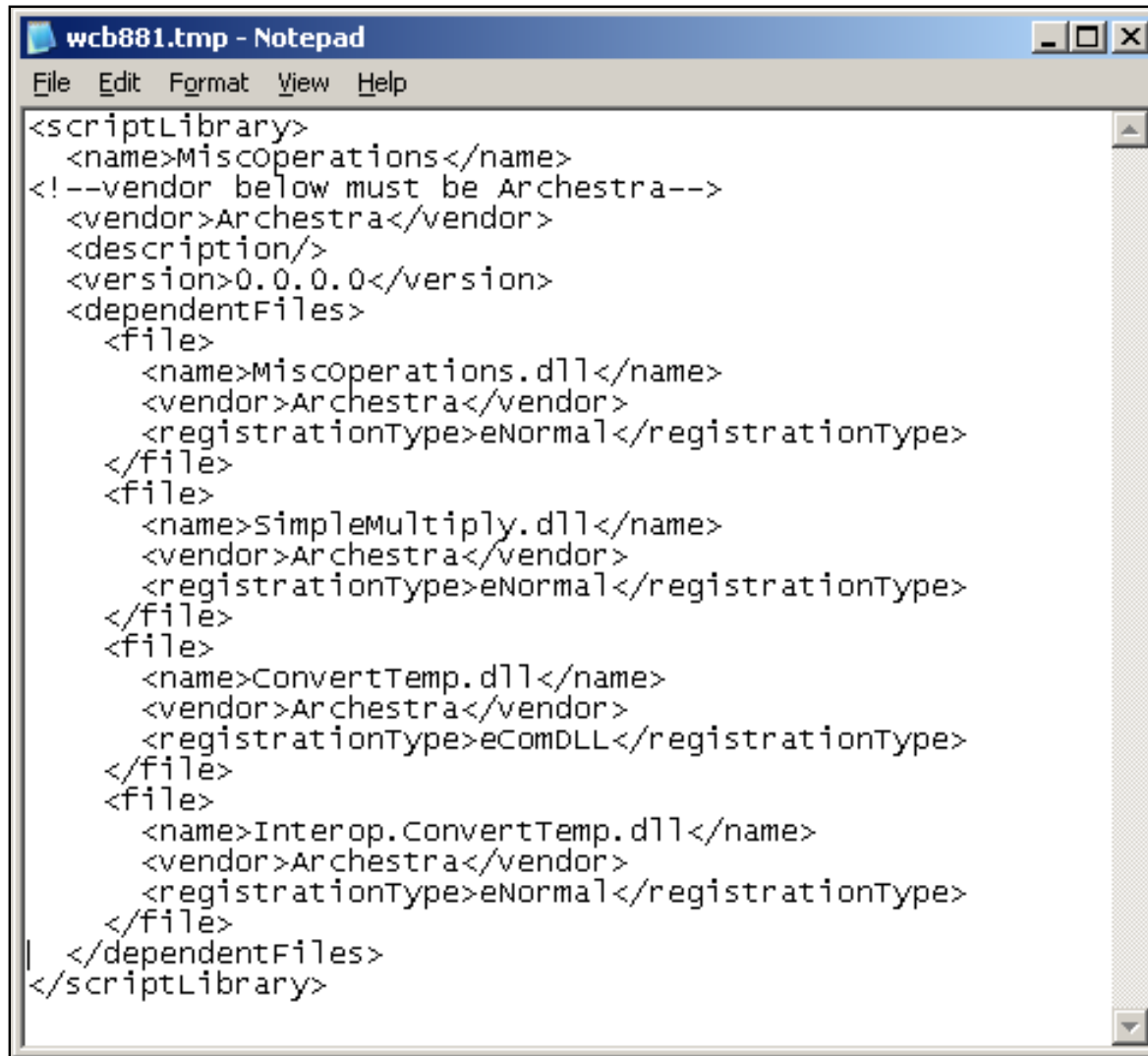
Note: The root node in this .xml file is <WWCabinetFileList>. Under this root are <FileItem> xml nodes. Each node provides a mapping via the **FileCode** xml attribute and the **FileName** xml attribute.

- o Locate the **FileItem** element with the FileName = MiscOperations.xml and note the FileCode. In this case it is: **wcb881.tmp**
- o Open the **wcb881.tmp** file using Notepad. The file content is shown below:

Figure 6: wcb881.tmp File Opened Using NotePad

- o Under the <dependentFiles> xml node, locate the <file> xml entry. For each dependent file, add

a corresponding <file> xml entry. Note that for the COM dll requiring registration, the <registrationType> element should be **eComDLL**. The modified file in this example looks like following figure:

A screenshot of a Notepad window titled "wcb881.tmp - Notepad". The window contains XML code for a script library. The code is as follows:

```
<scriptLibrary>
  <name>MiscOperations</name>
  <!--vendor below must be Archestra-->
  <vendor>Archestra</vendor>
  <description/>
  <version>0.0.0.0</version>
  <dependentFiles>
    <file>
      <name>MiscOperations.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eNormal</registrationType>
    </file>
    <file>
      <name>SimpleMultiply.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eNormal</registrationType>
    </file>
    <file>
      <name>ConvertTemp.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eComDLL</registrationType>
    </file>
    <file>
      <name>Interop.ConvertTemp.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eNormal</registrationType>
    </file>
  </dependentFiles>
</scriptLibrary>
```

Figure 7: Modified XML Nodes

- Save and close the file.
- Copy the .NET Class Library and its dependent files to the **working** folder (Figure 8 below).

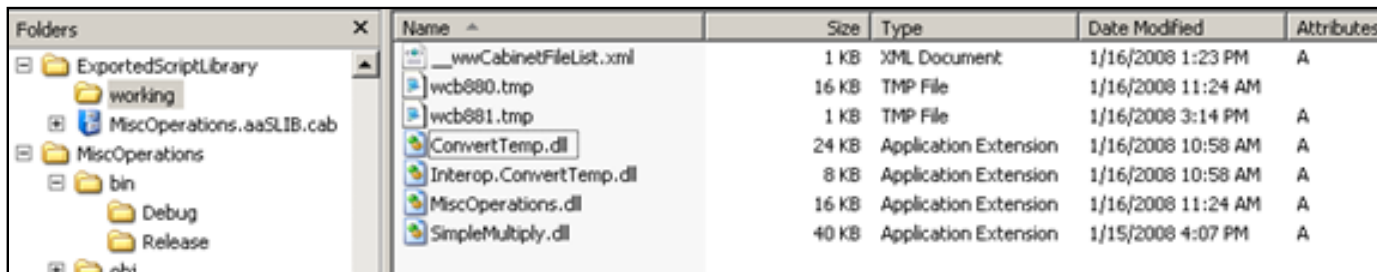


Figure 8: Copy files to the Working File

- o Open a Command Prompt window and navigate to the **working** folder.
- o Type `Cabarc N MiscOperations.aaSSLIB *.*` as shown in Figure 9 (below):

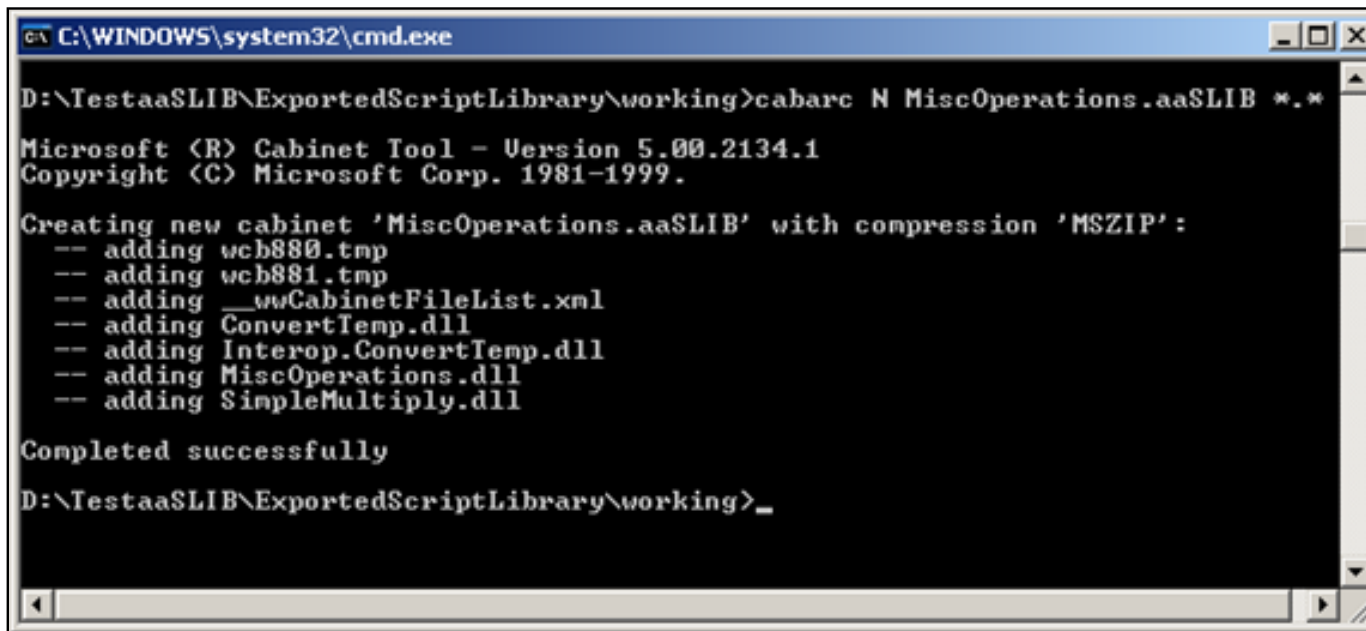


Figure 9: Command Prompt

Note: Download the .cab file utility, cabarc.exe, from the Microsoft website. The utility must be copied into your \system32 folder.

The resulting **MiscOperations.aaSSLIB** file can now be imported into the Galaxy. The script function library is automatically deployed with its dependent files and any relevant COM dlls are registered.

3. Test this library to see if the error **Unable to load DLL** is resolved.

Scenario 2 - Implementing a Modified .NET Class Library

The .NET Class Library is imported and deployed in ArcestrA IDE. You decide to modify the .NET Class Library by adding a new method. For this example, it is called **MiscOperations.dll**.

After rebuilding the new .NET class library, it is packaged once again in the .aaSLIB file as described in the steps above. The .NET Class library is re-imported, but the newly-added method does not appear in the Script Function Browser and it cannot be used. It looks like the old dll is still in effect.

How do I get the newly rebuilt .NET Class library into effect?

Solution

Increment the version of the .NET Class Library and repackage it in the .aaSLIB file after making changes in the .NET Class Library.

To reimport the .aaSLIB file

1. Update the version data in the .NET Class Library. Do this by incrementing the version number in the **AssemblyVersion** attribute. For example, if this library is developed in C#, then increment the version number in the "AssemblyVersion" attribute, in the **AssemblyInfo.cs** file.
2. Rebuild the .NET Class library.
3. Copy the new .NET Class Library in the Working folder as described in the [step above](#).
4. Replace the old .tmp file with a newly created .tmp file. This step can be best explained by using the same example of .NET Class Library as mentioned above.

Using [the previous step \(above\)](#), note the contents of the __wwCabinetFileList.xml file.

```
<FileItem FileName="MiscOperations.dll" FileCode="wcb880.tmp" />
```

In this example, **MiscOperations.dll** is the .NET class library for which there is a newer version. The above line in the xml file **__wwCabinetFileList.xml** indicates that the file **MiscOperations.dll** has the filecode **wcb880.tmp**.

Also note that the file **wcb880.tmp** exists in the working folder. This wcb880.tmp file is actually the same as MiscOperations.dll. So when a newer version of MiscOperations.dll is desired, the corresponding .tmp file should also be newly-created.

To do this, simply make a copy of **MiscOperations.dll** in the Working folder and rename it to **wcb880.tmp**.

5. Now run the cabarc utility as shown in [previous steps](#).
6. Import the new .aaSLIB file and test to see if this is the newer version of the .NET Class Library.

Scenario 3 - Automatically Copying Dependent Files

This Scenario includes the following elements (Figure 10 below):

The .NET Class Library **Miscoperations.dll** uses a COM dll called **ConvertTemp.dll**.

This COM dll has another dependent file called **MakeTempFilenameDLL.dll**, which is a simple Win32 dll with exported functions.

How do I package all the dependent files so that they are automatically copied to correct Arcestra directories when the .NET Class library is imported and deployed?

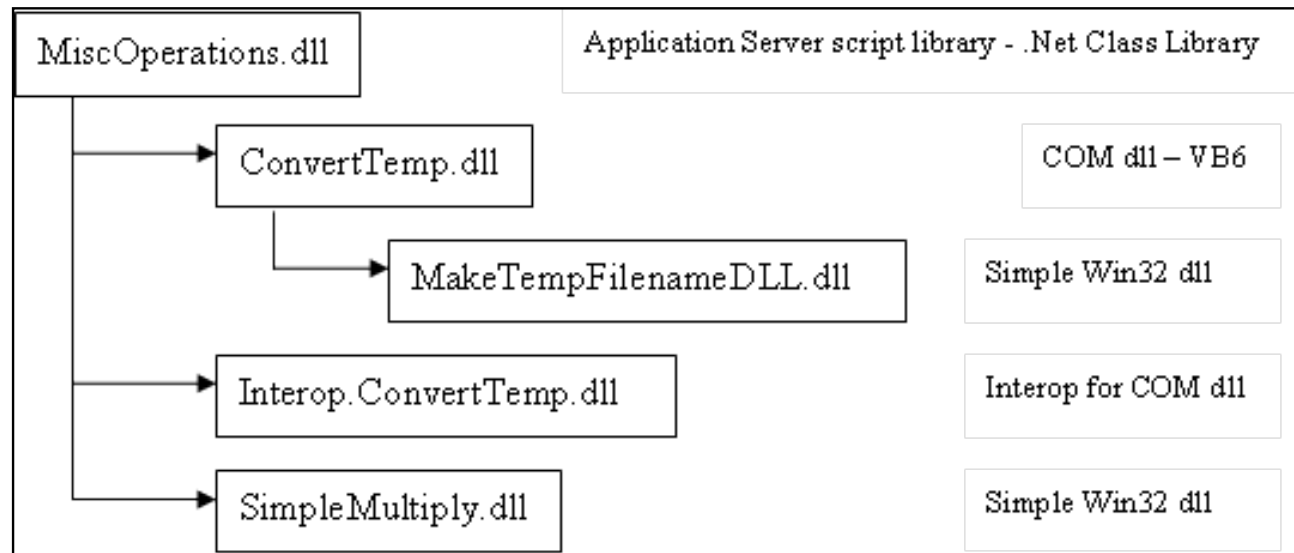


Figure 10: Scenario Three Class Library List

Solution

If a COM library developed with Visual Studio 6 or earlier has file dependencies, ensure that you include those files in the **.aaSLIB** file and include a `<file> ... </file>` entry in the .xml file under the `<dependentFiles>` node for

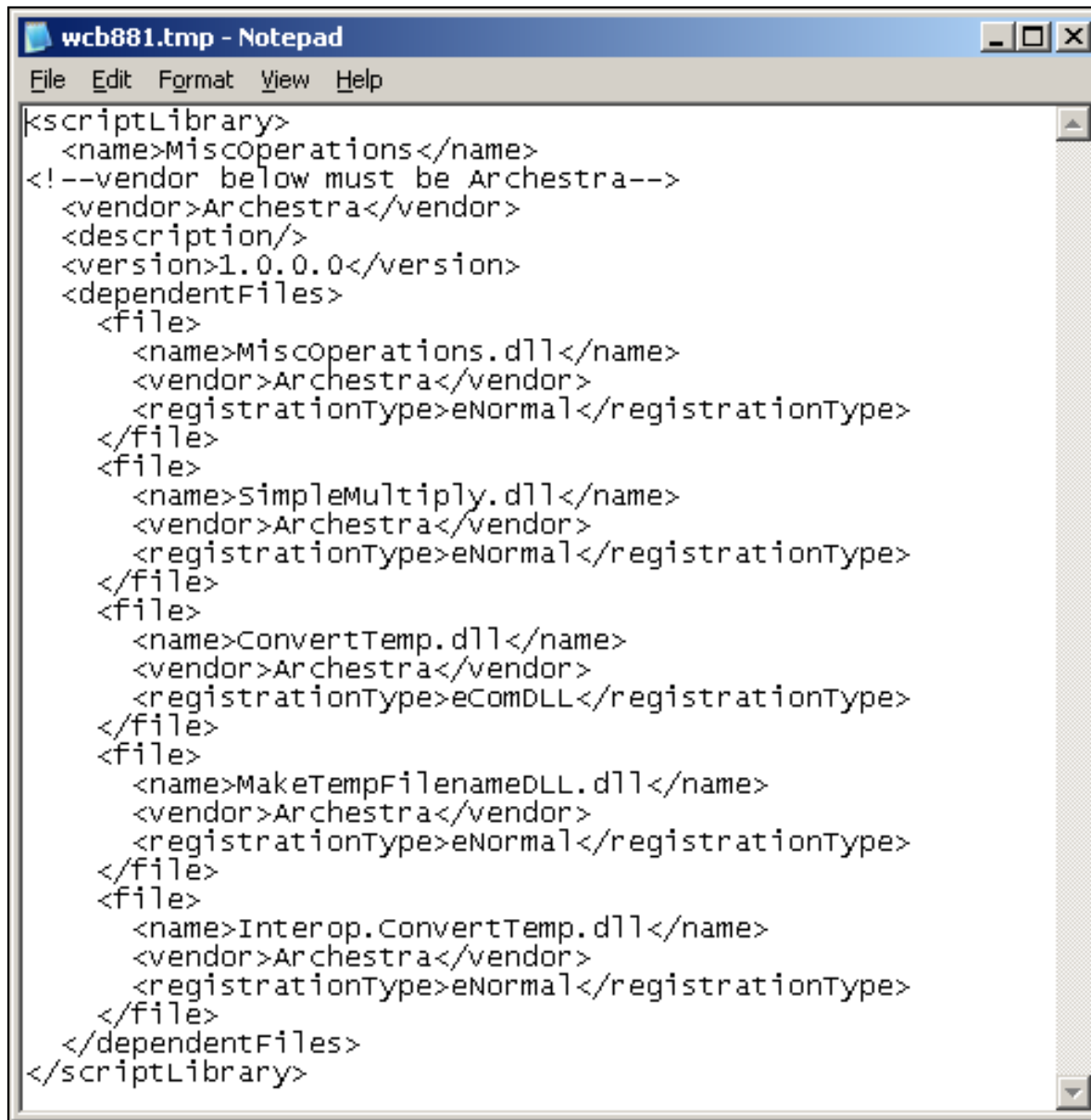
each dependent file. Otherwise, objects that use the script function library will not be deployed properly.

To package dependent files of any COM library used in the .NET Class Library

With reference to the example presented in Figure 10 (above):

- **Miscoperations.dll** is a .NET Class Library.
- **ConvertTemp.dll** is a COM library that is used in the .NET Class Library.
- **MakeTempFilenameDLL.dll** is a simple Win32 dll that is used in the ConvertTemp.dll COM dll.

In this case the XML file entries should be modified as follows:



```
wcb881.tmp - Notepad
File Edit Format View Help
<scriptLibrary>
  <name>MiscOperations</name>
  <!--vendor below must be Archestra-->
  <vendor>Archestra</vendor>
  <description/>
  <version>1.0.0.0</version>
  <dependentFiles>
    <file>
      <name>MiscOperations.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eNormal</registrationType>
    </file>
    <file>
      <name>SimpleMultiply.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eNormal</registrationType>
    </file>
    <file>
      <name>ConvertTemp.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eComDLL</registrationType>
    </file>
    <file>
      <name>MakeTempFilenameDLL.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eNormal</registrationType>
    </file>
    <file>
      <name>Interop.ConvertTemp.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eNormal</registrationType>
    </file>
  </dependentFiles>
</scriptLibrary>
```

Figure 11: Add <file> Entries For Each Dependent DLL in Notepad

Scenario 4 - Working with Third Party Methods and Multiple Dependencies

The .NET Class Library **Miscoperations.dll** uses methods exposed by a third party software product. This product consists of many dependent files and the vendor has provided a Merge Module (.msm file) for redistributing the dependent files.

How do I package the Merge Module so that all the required dependent files are automatically copied to correct ArchestrA directories when the .NET Class library is imported and deployed?

Solution

- Modify the **__wwCabinetFileList.xml** file to include the Merge Module provided by the vendor.
- Modify the .tmp file corresponding to the .xml Filename attribute.

If a COM library developed with Visual Studio 6 or earlier has file dependencies, ensure that you include those files in the .aaSLIB file and include a `<file> ... </file>` entry in the .xml file under `<dependentFiles>` for each dependent file. Otherwise, objects that use the script function library will not be deployed properly.

To package the Merge Module provided by the Vendor in an .aaSLIB file

Example: The .NET class library **Miscoperations.dll** uses the files provided in the **ThirdParty.msm** merge module.

1. Include the Merge Module in the **__wwCabinetFileList.xml** file as follows:

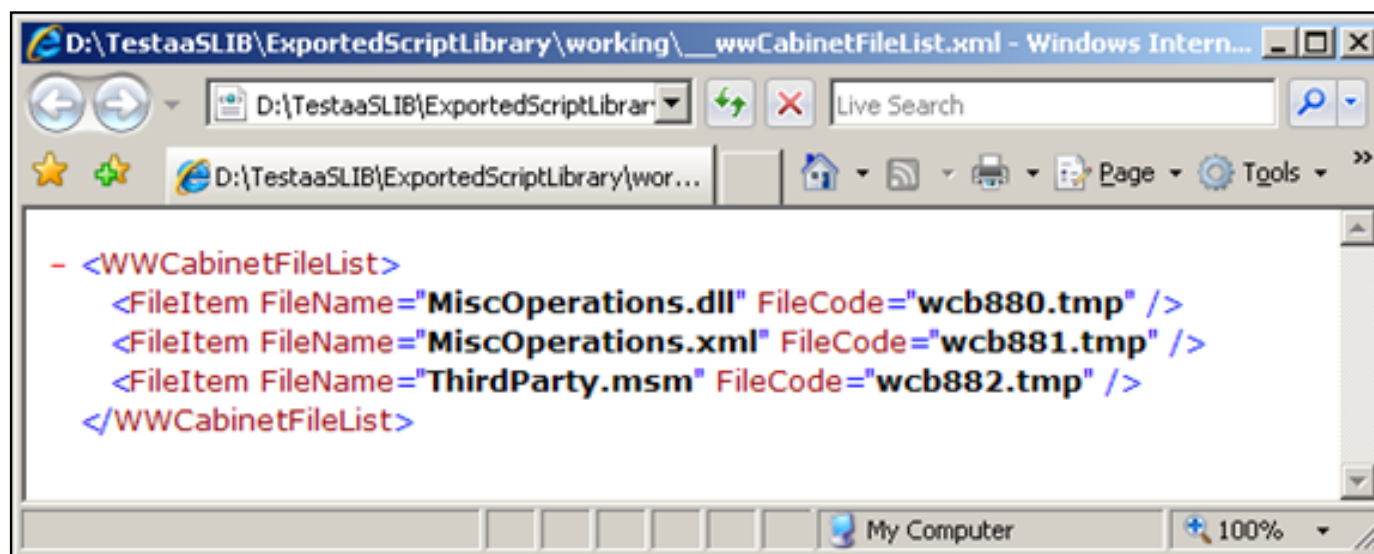
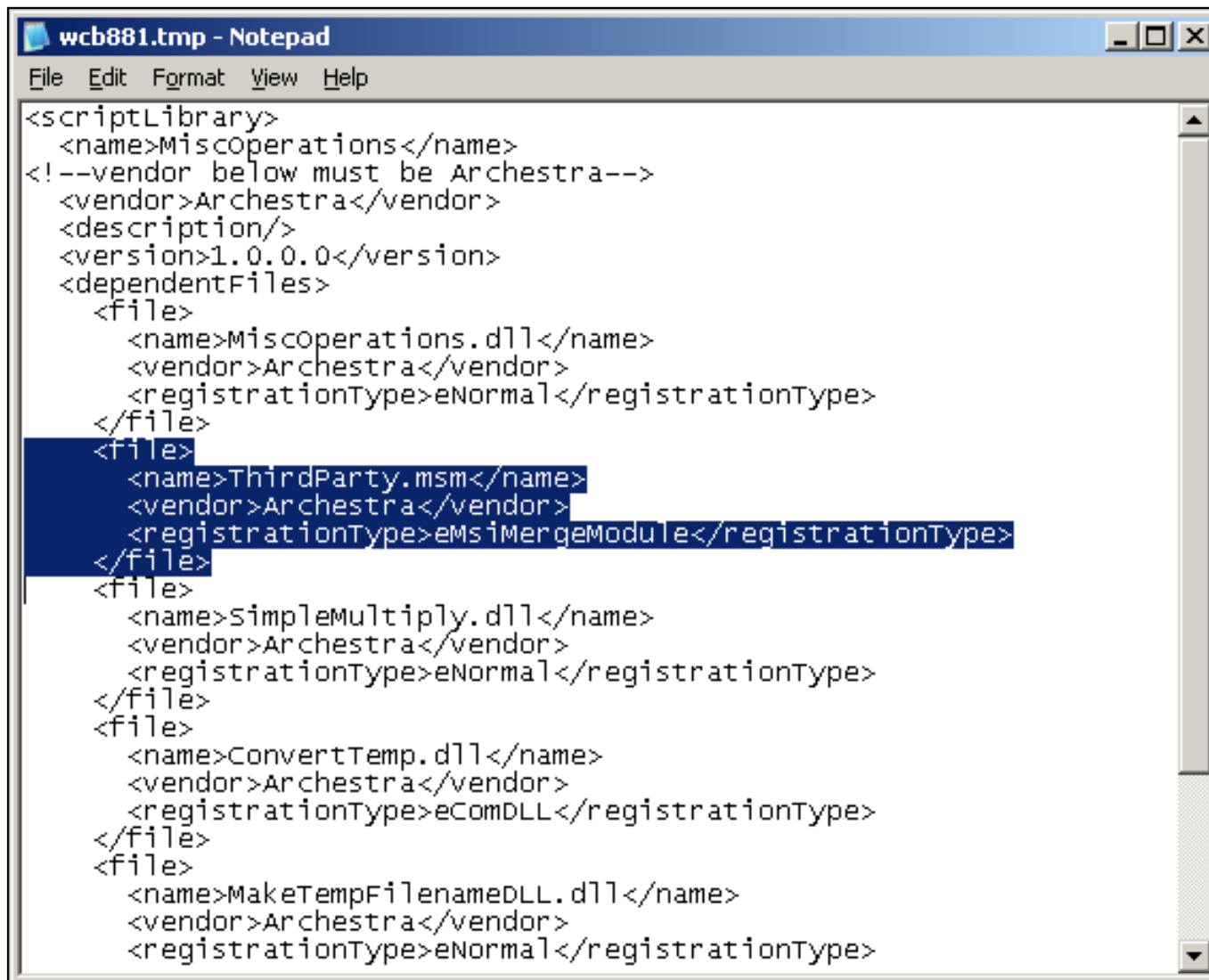


Figure 12: Include the Merge Module in the FileList Node

2. Modify the .tmp file- wcb881.tmp corresponding to the MiscOperations.xml Filename attribute as follows. Note the `<registrationType>` is **eMsiMergeModule**:



```
wcb881.tmp - Notepad
File Edit Format View Help
<scriptLibrary>
  <name>MiscOperations</name>
  <!--vendor below must be Archestra-->
  <vendor>Archestra</vendor>
  <description/>
  <version>1.0.0.0</version>
  <dependentFiles>
    <file>
      <name>MiscOperations.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eNormal</registrationType>
    </file>
    <file>
      <name>ThirdParty.msm</name>
      <vendor>Archestra</vendor>
      <registrationType>eMSiMergeModule</registrationType>
    </file>
    <file>
      <name>SimpleMultiply.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eNormal</registrationType>
    </file>
    <file>
      <name>ConvertTemp.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eComDLL</registrationType>
    </file>
    <file>
      <name>MakeTempFilenameDLL.dll</name>
      <vendor>Archestra</vendor>
      <registrationType>eNormal</registrationType>
    </file>
  </dependentFiles>
</scriptLibrary>
```

Figure 13: Modify the .tmp File in Notepad

3. Copy the ThirdParty.msm as wcb882.tmp to the working folder.
4. Run the cabarc utility to package the MiscOperations.aaSLIB file.

P. Kulkarni

For technical support questions, send an e-mail to support@wonderware.com.



[Back to top](#)

©2008 Invensys Systems, Inc. All rights reserved. No part of the material protected by this copyright may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, broadcasting, or by any information storage and retrieval system, without permission in writing from Invensys Systems, Inc. [Terms of Use](#).