Tech Note 569

Using Merge Modules

Topic#: 002321
Created: January 2009

Introduction

This *Tech Note* describes using Merge Modules in the context of importing custom DLLs and any related DLLs into your Application Server Galaxy.

Wonderware Application Server provides you the ability to leverage the .NET development language for custom and advanced functionality made available through the ArchestrA Integrated Development Environment (IDE).

Custom Script Function Libraries are integrated with Wonderware Application Server by

- Importing multiple, independent DLLs, or;
- Importing multiple DLLs that have dependencies into the Galaxy.

Wonderware Application Server accommodates importing of multiple DLLs and DLLs with dependencies through the use of **aaSLIB** files. **aaSLIB** files make managing multiple DLLs and their dependencies easier by eliminating the need to track multiple DLLs.

Working Definitions

This *Tech Note* uses terms that may be familiar but which can be used in different ways. The following definitions are applied in the context of this document:

- A **Custom DLL** (Shared Library) is simply compiled code that can be accessed from scripting or from a .NET control. A

DLL is a module that contains functions and data that can be used by another module (application or DLL).

It can also be described as a collection of small programs, any of which can be called when needed by a larger program that is running in the computer. The small program that lets the larger program communicate with a specific device such as a printer or scanner is often packaged as a DLL program (usually referred to as a DLL file). DLL files that support specific device operation are known as device drivers.

Source Citation

In this context, the custom DLLs provide added functionality for use by Wonderware Application Server.

Note: The Custom DLL is not to be confused with a .NET Client Control. The import and management of .NET Client Controls is different, and will be addressed in a future *Tech Note*.

Application Versions

- Wonderware Application Server 3.0 and later.

Summary Workflow

The following steps outline best practice for ensuring all script function libraries are correctly managed in the Galaxy.

1. Document all the DLLs used in the Galaxy in a table format.
2. For each DLL, identify if it is either a top-level DLL or a dependent DLL or both in different scenarios as the case may be.

Note: If the dependencies between DLLs is not explicitly known, the **.NET Reflector** tool can be used to discover them. This free tool can be downloaded from **Red Gate software**.

3. For each top-level DLL, identify the dependent DLLs that are needed for the top-level DLL to function correctly. These top-level DLLs with dependencies will determine the number of aaSLIB files that will need to be created.

4. Create **aaSLIB** files for each top-level DLL. Make sure you correctly reference the dependent DLLs that were noted during the aaSLIB creation process described in **Tech Note 566 Ensuring Accurate and Consistent Deployment of Custom Script Function Libraries**.

   Only the top-level DLL defined in the aaSLIB package will have all of its exposed methods available from the Script Function Browser during script development.

Note: For details about creating and referencing dependent DLLs, see **Tech Note 518 - Managing Custom Script Function Libraries for Use in Application Server** or alternatively refer to the section in this document called Working with aaSLIB Files.

5. Import one-by-one each stand-alone DLL and each aaSLIB file into the Galaxy using the **Import/Script Function Library** main menu commands in the IDE.

6. Once all the needed stand-alone DLLs or aaSLIB files have been imported, ensure to validate all objects that contain script references to the imported script function libraries. This step allows verification that script references to the imported script function libraries resolve correctly.

Note: If you are having trouble getting an object to validate correctly immediately after importing a script function library, try rebooting the GR Node.

7. Deploy the objects that contain the script references to the imported script function libraries.

8. Test & debug the scripts to ensure expected behavior.

Using Merge Modules

If you find that a set of DLLs (and their dependencies) are needed several times across multiple aaSLIB files, or that you need to reference a set of DLLs from a third party, you can create a Microsoft Merge Module (*.msm) file to package the DLLs into a single manageable file.

About Merge Modules (*.msm)

A merge module is a special kind of Windows Installer database that contains the components needed to install a discrete software bundle. A merge module cannot be installed alone, but must be merged into a standard Windows Installer installation during the creation of the installation. Typically, a merge module (or a merge module collection related by dependencies) installs a software product or portion of a product at runtime. Courtesy of **Wikipedia**.

Using merge modules you can add self-contained software modules to multiple installations.

For example, if there are a number of applications that require a specifically configured component, it would be possible to create a merge module that installs and configures that component. That merge module could then be added to the installation packages of each product that required that particular component. This saves the effort of having to individually add the necessary files, registry entries, and other components to every installation. It also saves time if updates are needed, as instead of updating the installations for all applications, only the merge module is updated, and only the installations need to be rebuilt.

In this context, when you create the aaSLIB file, instead of referencing several dependent DLLs, you can reference the single dependent *.msm package that contains the dependent DLLs.

This makes for a more modular and manageable aaSLIB file.

Note: You must add the *.msm reference for each aaSLIB file as a dependent file.

Merge Modules can be created using Microsoft Visual Studio.

For more information about Merge Modules in this context, see **Tech Note 518 - Managing Custom Script Function Libraries for Use in Application Server**.

References

- ***Tech Note* 566 Ensuring Accurate and Consistent Deployment of Custom Script Function Libraries**
- ***Tech Note* 568 How Wonderware Application Server Manages DLLs and aaSLIB Imports**
- **FAQ 38 Importing Custom DLLs**

S. Kermani

Back to top