Microsoft Enterprise Library 5.0 and Unity 2.0 Migration Guide

April 2010

Microsoft®
patterns & practices
proven practices for predictable results

Microsoft Enterprise Library 5.0 and Unity 2.0 Migration Guide

Contents

Introduction

Migrating your existing applications to use the latest versions of Enterprise Library and Unity can provide a range of benefits. These include:

- **Additional functionality**. New and upgraded features of the application blocks and the Unity dependency injection and interception mechanism can make future updates to your application easier and extend its capabilities.
- **Bug fixes**. New releases of Enterprise Library and Unity contain fixes for existing issues that were discovered after release of the previous versions.
- **Increased performance**. Many of the features of the new versions of Enterprise Library and Unity have been fine tuned to maximize performance.

- **Easier configuration**. Simplified schema syntax for Unity, and a new graphical configuration tool for both Enterprise Library and Unity (provided as both standalone and Visual Studio integrated versions) makes configuration easier and quicker.

This guide explains the opportunities open to you for migrating applications built using Enterprise Library versions 3.1, 4.0, and 4.1, and versions 1.0 and 1.1 of Unity to use version 5.0 or Enterprise Library and version 2.0 of Unity.

Because individual application scenarios and environments vary, and the way Enterprise Library and Unity are used within existing applications will differ considerably, this guide cannot guarantee success in every situation. However, it contains practical guidance that is based on knowledge gathered during the development of Enterprise Library 5.0, and through test migrations of a range of different existing applications. These applications included:

- The Enterprise Library 4.1 Hands-On Labs.
- A financial application that uses the Enterprise Library 4.0 Exception Handling and Logging blocks.
- A range of sample applications from presentations prepared by the Microsoft patterns & practices group. Together, these applications use all of the Enterprise Library 3.1 and 4.x application blocks.

The Enterprise Library developer community Web sites on CodePlex provide a forum for a wide range of topics, including migration, and are a good place to post details of any specific issues you encounter. The forum is actively monitored by the Enterprise Library sustained engineering team, which provides support for Enterprise Library and Unity.

Enterprise Library community site: http://www.codeplex.com/entlib/.

Unity community site: http://www.codeplex.com/unity/.

Migration and Update Scenarios

This guidance describes simple migration of applications from earlier versions of Enterprise Library and Unity to the current versions, and does not explore rewriting or adapting an existing application to use a different architectural style or new features. The basic premise is to replace the existing earlier version assemblies with the assemblies for Enterprise Library 5.0 and Unity 2.0 without re-architecting the application, re-writing the code, or requiring anything other than the simplest changes to configuration files and assembly references.

However, after you successfully migrate your application to use the new assemblies, you should consider whether to take this opportunity to upgrade your architecture style or code to take advantage of current application design principles and new Enterprise Library and Unity features.

For example, you should consider whether to adopt dependency injection for creating objects and obtaining references to objects, in both Enterprise Library and Unity. You may also want to take advantage of new features implemented in Enterprise Library and Unity to simplify your code or improve performance.

For more information on the changes to Enterprise Library and Unity, see the topic "*Changes in This Release*" in the installed documentation, and available online at http://go.microsoft.com/fwlink/?LinkId=188874.

Migration Prerequisites

There are some scenarios where upgrading from a previous version of Enterprise Library or Unity is not possible, or may require additional changes to your application code.

If you have modified the source code for Enterprise Library or Unity and recompiled it, your modifications will be lost when you migrate to version 5.0 of Enterprise Library and version 2.0 of Unity. The steps described in this guide for migrating applications assume that you are using the Microsoft signed assemblies provided with previous versions of Enterprise Library and Unity.

However, if you have recompiled the original source code without modifying it (perhaps to use the assemblies in partial trust scenarios in version 3.1, or to sign them with your own key), you will be able migrate successfully to version 5.0 providing that you can meet the following prerequisites:

- Windows Management Instrumentation (WMI) events are no longer exposed by Enterprise Library, and the Manageable Configuration Source no longer exposes the application configuration through WMI. The only WMI feature remaining in version 5.0 is the WMI Logging Trace Listener. If your application relies on WMI events or the WMI configuration features of Enterprise Library, consider removing this functionality if this is feasible. If your application is dependent on WMI events or WMI configuration, you cannot migrate it to Enterprise Library 5.0.
- Changes to the Policy Injection Application Block mean that existing applications that use this block cannot be upgraded directly to version 5.0 of Enterprise Library. At minimum you must change namespace references for the call handlers, which now reside in other blocks. The Caching Call Handler has also been removed from this release. If you need to use this call handler, you must download it from the Enterprise Library Web site on CodePlex at http://www.codeplex.com/entlib/. In addition, if you have created custom call handlers or matching rules, you will not be able to migrate your application to version 5.0 of Enterprise Library without additional work updating these. This topic is out of scope for this guide.
- Enterprise Library 5.0 uses a new configuration console and Visual Studio integrated configuration editor, and the configuration mechanism in Enterprise Library has changed in version 5.0. Custom providers that *do not* have a design-time experience (providers that are added to the configuration using the built-in "Add Custom ... Provider" option and that accept a name/value pair containing the configuration values) will continue to work in Enterprise Library

5.0. However, custom providers that have a design-time experience based on a previous version of the configuration tool are not compatible with Enterprise Library 5.0. Although you can adapt them to work with Enterprise Library 5.0, this migration path is out of scope for this guide. For more information about custom providers, see the topics in the section "Extending and Modifying Enterprise Library" in the documentation installed with Enterprise Library, and available online at http://go.microsoft.com/fwlink/?LinkId=188874.

- The configuration system and XML schema used by Unity have changed in version 2.0. You can upgrade existing Unity configuration files to the new schema, but you must factor in this work when you migrate an existing application that uses a previous version of Unity.
- Unity container extensions created for version 1.x are not compatible with Unity version 2.0. Although you can adapt them to work with Unity 2.0, this migration path is out of scope for this guide. For more information about Unity container extensions, see http://www.codeplex.com/unity/.

Suggested Steps for Migration

The following steps will help you to migrate your application to Enterprise Library 5.0 and Unity 2.0:

- Before You Begin
- Start the Migration
- Replace the Assemblies and References
- Update Configuration Files
- Deal with Breaking Changes and Deprecated Functionality
- Finalize the Migration

Steps that are specific to migrating from Enterprise Library version 4.0 are marked with **[*4.0*]**. You can omit these steps if you are migrating from version 4.1.

Steps that are specific to migrating from Enterprise Library version 3.1 are marked with **[*3.1*]**. You can omit these steps if you are migrating from version 4.0 or 4.1.

Before You Begin

- Review the most recently published lists of breaking changes and known issues for both Enterprise Library 5.0 and Unity 2.0. You will find these lists in the Release Notes documents available at http://www.codeplex.com/entlib/ and http://www.codeplex.com/unity/. You must make a decision as to whether you can adapt your application to work after reviewing these changes. If you can, make a note of the changes that are applicable so that you can update your application code after you finish migration. If you cannot work around these changes, you should not continue with the migration.

- [*4.0*] If you are upgrading from version 4.0 of Enterprise Library, you should also review the lists of breaking changes for version 4.1 of Enterprise Library.
- [*3.1*] If you are upgrading from version 3.1 of Enterprise Library, you should also review the lists of breaking changes for versions 4.0 and 4.1 of Enterprise Library.
  - Changes in Enterprise Library 4.0: http://msdn.microsoft.com/en-us/library/cc511712.aspx.
  - Changes in Enterprise Library 4.1 and Unity 1.2: http://msdn.microsoft.com/en-us/library/dd139937.aspx.

Start the Migration

- Confirm that the application solution currently compiles without errors and the application runs correctly. Confirm that all unit tests pass. This will ensure that any issues you encounter during the migration can be attributed to the migration, and not to an existing fault in the application.
- If any of the sections in your configuration files are encrypted, you must decrypt them using the tools provided with the version of Enterprise Library you are migrating from. After you complete the migration to version 5.0, you can reencrypt them using the version 5.0 configuration tools.

Replace the Assemblies and References

- Within your Visual Studio solution, update the Enterprise Library assembly references to point to the Enterprise Library version 5.0 assemblies. Typically, the Enterprise Library version 5.0 assemblies are installed in the %**Program Files%\Microsoft Enterprise Library 5.0 - April 2010\bin** folder. Follow these steps for every project in your solution:
  - Remove all existing references to Microsoft.Practices.EnterpriseLibrary assemblies.
  - If you are using assemblies located in your **lib** or **bin** folder, replace these with the version 5.0 assemblies located in the **bin** folder of your Enterprise Library installation.
  - Add references to your project for the Enterprise Library 5.0 version of the assemblies you require.
- If you are migrating an application that uses Unity, update the references to point to the Unity version 2.0 assemblies. Follow these steps for every project in your solution:
  - Remove all existing references to Microsoft.Practices.Unity assemblies.
  - If you are using assemblies located in your **lib** or **bin** folder, replace these with the Unity version 2.0 assemblies. All of these are located in the **bin** folder of your Unity installation.

    - Add references to your project for the Unity 2.0 version of the assemblies you require.

- Remove all references to the Object Builder assembly. All of this functionality is now included within the Unity assemblies. Follow these steps for every project in your solution that references the Object Builder assembly:
    - Remove the reference to Microsoft.Practices.ObjectBuild2.dll.
    - If you are using assemblies located in your **lib** or **bin** folder, delete the Microsoft.Practices.ObjectBuild2.dll assembly file from this folder.
- If your application uses the Policy Injection Application Block, you must add a reference to the container Service Location assembly. Follow these steps for every project in your solution that references the Policy Injection Application Block:
    - If you are using assemblies located in your **lib** or **bin** folder, copy the assembly Microsoft.Practices.ServiceLocation.dll from the **bin** folder of your Enterprise Library installation into your **lib** or **bin** folder.
    - Add a reference to the assembly Microsoft.Practices.ServiceLocation.dll to your project.
- If your application uses the Validation Application Block, you must add a reference to the .NET Data Annotations assembly. This assembly is provided with version 3.5 and later of the .NET Framework, which is installed automatically with Visual Studio 2008 and later. For every project in your solution that references the Validation Application Block:
    - Add a reference to the assembly System.ComponentModel.DataAnnotations.dll to your project.
- Ensure that your projects reference all of the mandatory Enterprise Library assemblies. For every project in your solution that uses Enterprise Library:
    - If you are using assemblies located in your **lib** or **bin** folder, copy the following assemblies from the **bin** folder of your Enterprise Library installation into your **lib** or **bin** folder if they are not already located there:
        - Microsoft.Practices.EnterpriseLibrary.Common.dll
        - Microsoft.Practices.Unity.dll
        - Microsoft.Practices.Unity.Interception.dll
        - Microsoft.Practices.ServiceLocation.dll
    - Add a reference to each of the assemblies listed above to your project if they are not already referenced.
- If you are using Unity as a dependency injection or interception mechanism in your application, ensure that your projects reference all of the mandatory Unity assemblies. For every project in your solution that uses Unity:

- If you are using assemblies located in your **lib** or **bin** folder, copy the following assemblies from **bin** folder of your Unity installation into your **lib** or **bin** folder if they are not already located there:
    - Microsoft.Practices.Unity.dll
    - Microsoft.Practices.Unity.Interception.dll
    - Microsoft.Practices.Unity.Configuration.dll
    - Microsoft.Practices.Unity.Interception.Configuration.dll
    - Microsoft.Practices.ServiceLocation.dll
- Add a reference to each of the assemblies listed above to your project if they are not already referenced.

Update Configuration Files

- Any configuration file that contains configuration information for Enterprise Library must be updated to refer to the current version of the assemblies. You must manually replace any reference to previous versions of these assemblies with the correct information for the release version, for every single fully qualified type name:
    - Use a text editor to completely remove the **Version**, **Culture**, and **PublicKeyToken** attributes from every Enterprise Library element that contains these attributes.
    - Open your configuration file in the Enterprise Library version 5.0 configuration editor and then save it. This will add the correct version, culture, and public key token values to each element.
- Any configuration file that contains configuration information for Unity must be updated to refer to the current version of the assemblies. However, the configuration schema for Unity has changed in version 2.0. If you have a **<unity>** section in any of your configuration files, you must follow these steps to update this section:
    - Use a text editor to completely remove the **Version**, **Culture**, and **PublicKeyToken** attributes from every element in the **<unity>** section that contains these attributes.
    - Enable the Enterprise Library and Unity XML schema in Visual Studio. To do this, open the configuration file in Visual Studio, open the **XML** menu, and click **Schemas**. In the XML Schemas dialog, locate the Unity schema (UnityConfiguration20.xsd) and change the value in the **Use** column to **Use this schema**. Then click **OK**.
    - Modify the content of the **<unity>** section of the configuration file so that it matches the new schema. The IntelliSense and AutoComplete features enabled by the XML schema will make this easier. For details of the new configuration schema for Unity, see "*Design Time Configuration*" in the documentation installed with Enterprise Library and Unity.

Deal With Breaking Changes and Deprecated Functionality

- If your application uses the Policy Injection Application Block, you must update your code to take account of the change in location of the call handlers. For every project in your solution that uses a policy injection call hander:
    - Determine the correct namespace for each call handler you use. The namespaces are:
        - **Authorization handler**: Microsoft.Practices.EnterpriseLibrary.Security.PolicyInjection
        - **Exception handling handler**: Microsoft.Practices.EnterpriseLibrary.ExceptionHandling.PolicyInjection
        - **Logging handler**: Microsoft.Practices.EnterpriseLibrary.Logging.PolicyInjection
        - **Validation handler**: Microsoft.Practices.EnterpriseLibrary.Validation.PolicyInjection
        - **Performance Counter handler**: This call handler remains in the same location as previous versions (Microsoft.Practices.EnterpriseLibrary.PolicyInjection.CallHandlers)
        - **Caching handler**: This handler is no longer included in Enterprise Library. You must download it from Enterprise Library Contrib Project if you need to use it in your application.
    - Ensure that your project references the assemblies containing the required namespaces. It is likely that this is already the case if you are using the application block that a call handler instantiates.
    - If you are *not* using the Performance Counters handler, remove any **using**, **Imports**, or **open** statements that refer to the namespace Microsoft.Practices.EnterpriseLibrary.PolicyInjection.CallHandlers.
    - In your code, ensure that you have a **using**, **Imports**, or **open** statement for the namespace of each call handler you use.
    - If you are using the Validation handler, add **using**, **Imports**, or **open** statements that refer to the namespace Microsoft.Practices.EnterpriseLibrary.Validation.Validators.
- Compile the solution and examine the Errors list for warnings and errors. If you encounter any errors, check the breaking changes lists to see if the error correlates with a known breaking change, and update the code to account for this change if possible. If you cannot accommodate the change, you may need to abandon the migration.
- If you encounter any warnings of deprecated functionality, update your code to use the new API as specified in the error message or the breaking changes lists.

- If you encounter an error or warning that is not documented in the breaking changes lists, and you are unable to find a solution, please post details to the Enterprise Library or Unity forums on CodePlex at http://www.codeplex.com/entlib/ or http://www.codeplex.com/unity/.

Finalize the Migration

- After successful compilation of the entire solution, review the lists that document changes in behavior of Enterprise Library and Unity to ensure that none affect your application. For example, Enterprise Library version 5.0 now raises an **ActivationException** when it encounters a configuration error, instead of the **ConfigurationErrorsException** raised in previous versions.
- Run the application and execute all of your unit tests to ensure that they all pass.
- Reencrypt the relevant sections of your configuration files as necessary using the version 5.0 configuration tools.