Mixed (native and managed) assemblies

Article • 08/03/2021

Mixed assemblies are capable of containing both unmanaged machine instructions and MSIL instructions. This allows them to call and be called by .NET components, while retaining compatibility with native C++ libraries. Using mixed assemblies, developers can author applications using a mixture of .NET and native C++ code.

For example, an existing library consisting entirely of native C++ code can be brought to the .NET platform by recompiling just one module with the **/clr** compiler switch. This module is then able to use .NET features, but remains compatible with the remainder of the application. It is even possible to decide between managed and native compilation on a function-by-function basis within the same file (see managed, unmanaged).

Visual C++ only supports the generation of mixed managed assemblies by using the **/clr** compiler option. The **/clr:pure** and **/clr:safe** compiler options are deprecated in Visual Studio 2015 and unsupported in Visual Studio 2017. If you require pure or verifiable managed assemblies, we recommend you create them by using C#.

Earlier versions of the Microsoft C++ compiler toolset supported the generation of three distinct types of managed assemblies: mixed, pure, and verifiable. The latter two are discussed in Pure and Verifiable Code (C++/CLI).

In this section

How to: Migrate to /clr
Describes the recommended steps for either introducing or upgrading .NET functionality in your application.

How to: Compile MFC and ATL Code By Using /clr
Discusses how to compile existing MFC and ATL programs to target the Common Language Runtime.

Initialization of Mixed Assemblies
Describes the "loader lock" problem and solutions.

Library Support for Mixed Assemblies
Discusses how to use native libraries in **/clr** compilations.

Performance Considerations
Describes the performance implications of mixed assemblies and data marshaling.

Application Domains and Visual C++
Discusses Visual C++ support for application domains.

Double Thunking
Discusses the performance implications of a native entry point for a managed function.

Avoiding Exceptions on CLR Shutdown When Consuming COM Objects Built with /clr
Discusses how to ensure proper shutdown of a managed application that consumes a COM object compiled with **/clr**.

How to: Create a Partially Trusted Application by Removing Dependency on the CRT Library DLL
Discusses how to create a partially trusted Common Language Runtime application using Visual C++ by removing dependency on msvcm90.dll.

For more information about coding guidelines for mixed assemblies, see An Overview of Managed/Unmanaged Code Interoperability.

See also

- Native and .NET Interoperability

Feedback