



Using ClearImage COM within .NET

Memory Garbage Collection

Introduction

The .NET framework introduced to windows, a new memory garbage collection method, which does not remove from memory the ClearImage COM objects that are no longer in use. When too many such objects remain in memory, internal resources may run low and ClearImage COM objects will return errors. This support note describes the problem in detail and offers a simple workaround to avoid the problem.

Problem Description

ClearImage COM memory allocation and release, in conjunction with the customer's application, are subject to the following constraints under the .NET environment:

- ClearImage COM object may allocate substantial amounts of memory, which is released when object is destroyed.
- Besides image memory, ClearImage requires a number of other, limited internal resources, which are also freed when the object is destroyed.
- VB 6 and C++ Smart Pointers destroy the objects when they exit the function scope, and either releasing memory and resources or setting them to **Nothing** or **NULL**.
- In .NET, Managed objects are not destroyed till later, when .NET decides to do garbage collection.

Consequently, it is possible for ClearImage to run out of internal resources before .NET invokes garbage collection.

Solution

This problem can be easily circumvented by forcing .NET to release memory resources belonging to objects that are no longer in use. Invoke the System Garbage Collection process outside of the functions or methods where ClearImage objects are created and referenced.

```
System.GC.Collect  
System.GC.WaitForPendingFinalizers
```

Notes

All ClearImage objects are derived from CiServer object. For most applications one instance of the CiServer object suffices. Declare CiServer in the global scope, rather than in subs or functions: `Dim Ci As New ClearImage.CiServer`

Building Assembly with Strong Names

Introduction

.NET introduced a new security mechanism through the use of strong names. When you compile an assembly with a strong name, any referenced assemblies must also have strong names. ClearImage COM has no strong name. When you try to compile a managed assembly that references ClearImage COM, you may receive the following error message:

Assembly generation failed -- Referenced assembly 'Interop.ClearImage' does not have a strong name

Solution

Microsoft Visual C# .NET

If your Visual C# .NET project references the COM Interop assembly, then the COM Interop assembly is generated for you when you reference the COM dynamic-link library (DLL). You can specify the wrapper assembly key file in the Visual C# project properties as follows:

1. In Microsoft Visual Studio .NET, open the properties of the Visual C# project in which you want to reference the COM component.
2. In the tree, click **Common Properties**, and then click **General**.
3. In the **Wrapper Assembly Key File** field, add the key file.
4. Rebuild the project.

Microsoft Visual Basic .NET

If your Visual Basic .NET project references the COM Interop assembly, you must manually generate the COM Interop assembly with **Tlbimp.exe**, as follows:

1. In Microsoft Visual Studio .NET, open the properties of the Visual Basic project in which you want to reference the COM component.
2. Delete the existing reference to your COM component.
3. Run **Tlbimp.exe** from a command prompt on your COM DLL to generate an Interop assembly that has a strong name, as follows:

```
tlbimp.exe ClearImage.dll /keyfile:KeyFile.snk /out:ClearImageInterop.dll
```

4. Add a reference to the output file from **Tlbimp.exe** to your Visual Basic .NET project.
5. Rebuild the project.

If the assembly is delay signed instead of fully signed, the wrapper assembly must be delay signed. To create an assembly that is delay signed, generate the wrapper by using the **TlbImp.exe** tool. To do this, follow the previous steps. If you use the "Microsoft Visual Basic .NET" steps, modify the command line that is provided in step 3 as follows:

```
tlbimp.exe ClearImage.dll /delaysign /publickey:PublicKeyFile.snk  
/out:ClearImageInterop.dll
```

Notes

If you already compiled assembly referencing ClearImage COM and now want to build signed assembly, you might need to:

1. Remove reference to ClearImage COM.
2. Add back in reference to signed ClearImageInterop.dll created by **tlbimp**

References

- <http://www.sampublishing.com/articles/article.asp?p=26993&redir=1&rl=1>