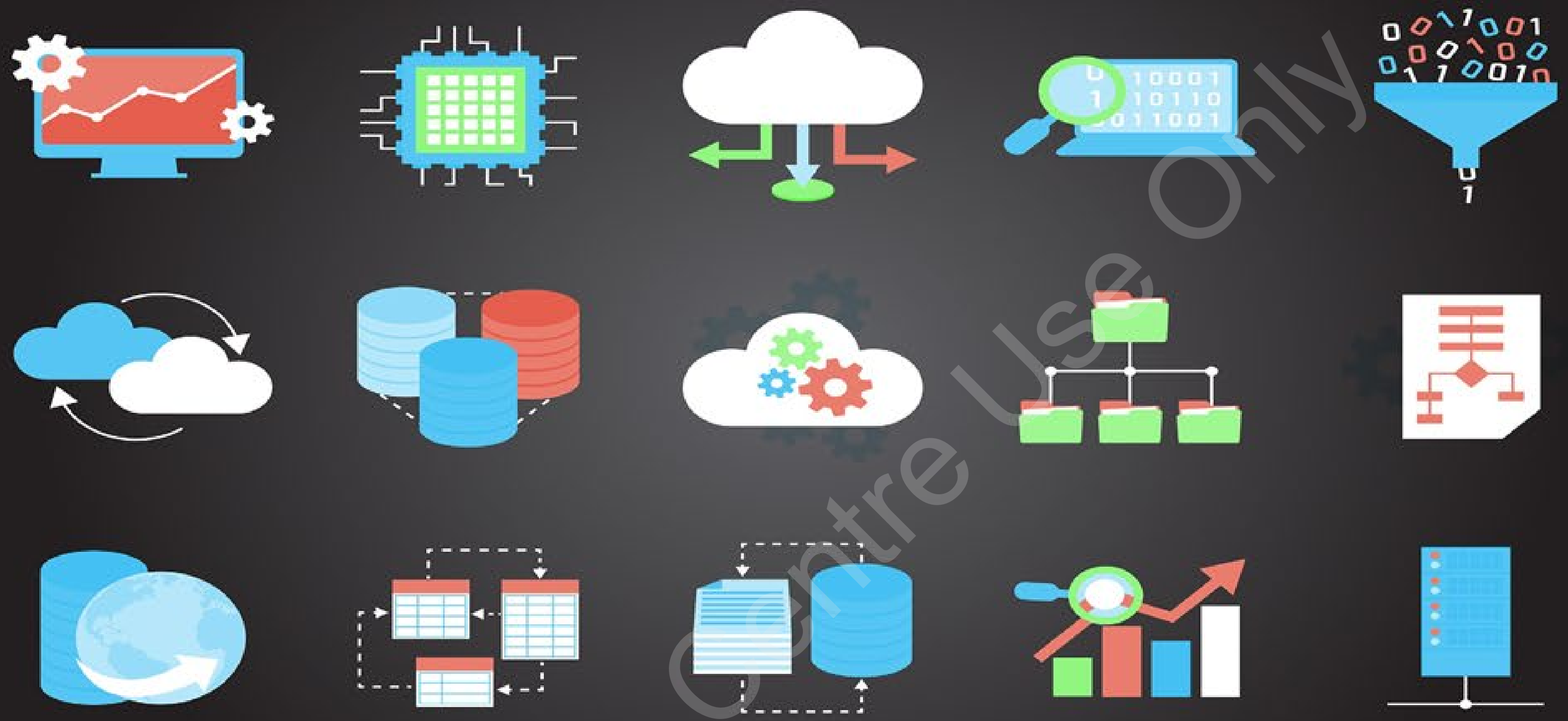


Session 15:
Working with Shared Assemblies

Session Objectives

- Describe Shared Assemblies
- Explain how to implement shared assemblies across multiple servers
- Explain how to deploy assemblies to global assembly cache
- Identify the implementation of assembly versioning
- Describe how to create an assembly manifest
- Describe how to configure assembly binding redirects

Introduction to Shared Assemblies [1-3]

An assembly is a single deployable unit which can be private or shared.

Assemblies perform as self-defining units, including complete data on class implementations, on structures as well as interfaces.

They are used for version control and promote reuse of code.

An assembly can include either .exe or .dll files.

Assemblies facilitate activation scoping as well as assigning permissions for security.

Assemblies comprise resources and types also store namespaces.

Introduction to Shared Assemblies [2-3]

Shared Assemblies in .NET Framework

- An assembly that can be used by multiple applications is called a shared assembly.

Global Assembly Cache (GAC)

- GAC is a central repository. Applications depending on a shared assembly kept in the GAC make use of the same disk files.
- Security policies and configuration applied to a shared assembly have an effect on all applications, thus offering an administrative advantage.

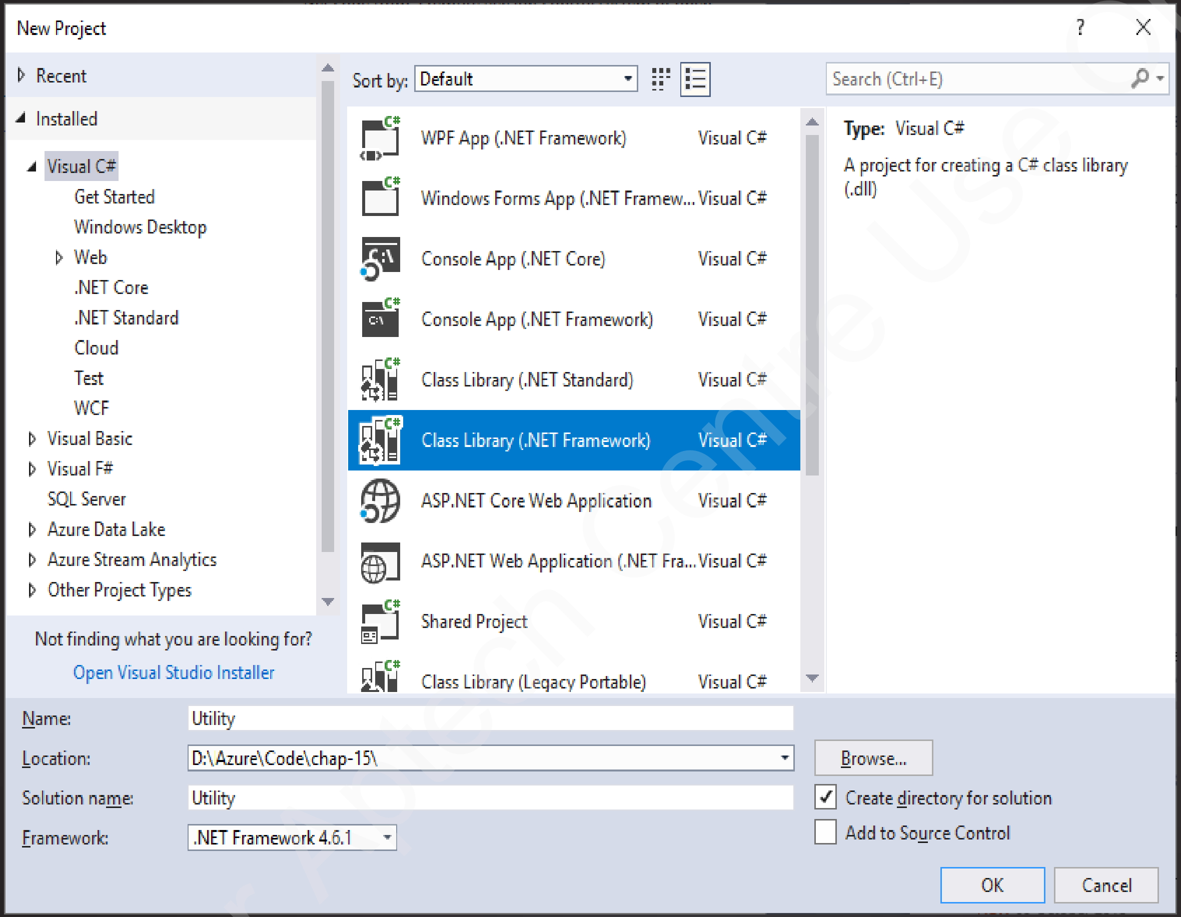
Assembly Properties

- Single file
- Multifile
- There are several options for grouping resources and code modules into assemblies such as version, deployment, reuse, security, and scope

Introduction to Shared Assemblies [3-3]

Version	Modules with the same version information can be grouped.
Deployment	Resources and code modules that support the available deployment model can be grouped.
Reuse	Modules that can function together in a logical manner for a given purpose can be grouped and reused.
Security	Modules that contain types which require same security permissions can be grouped.
Scope	Modules that contain types for which visibility is limited to the same assembly can be grouped.

Creating an Assembly [1-7]



New Project
Recent
Installed
Visual C#
Get Started
Windows Desktop
Web
.NET Core
.NET Standard
Cloud
Test
WCF
Visual Basic
Visual F#
SQL Server
Azure Data Lake
Azure Stream Analytics
Other Project Types
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
WPF App (.NET Framework) Visual C#
Windows Forms App (.NET Framew... Visual C#
Console App (.NET Core) Visual C#
Console App (.NET Framework) Visual C#
Class Library (.NET Standard) Visual C#
Class Library (.NET Framework) Visual C#
ASP.NET Core Web Application Visual C#
ASP.NET Web Application (.NET Fra... Visual C#
Shared Project Visual C#
Class Library (Legacy Portable) Visual C#
Browse...
OK
Cancel

Type: Visual C#

A project for creating a C# class library (.dll)

Name: Utility

Location: D:\Azure\Code\chap-15\

Solution name: Utility

Framework: .NET Framework 4.6.1

☑ Create directory for solution

☐ Add to Source Control

Creation of Class Library

Creating an Assembly [2-7]

```
using System;
namespace Utility
{ public class ConvertType
    {   public int StringToInt(string inputString)
        {  int output = 0;
// in case of any error in conversion this method will return
// default value 0
            try {output = Convert.ToInt32(inputString); }
            catch { }
            return output; }
}
}
```

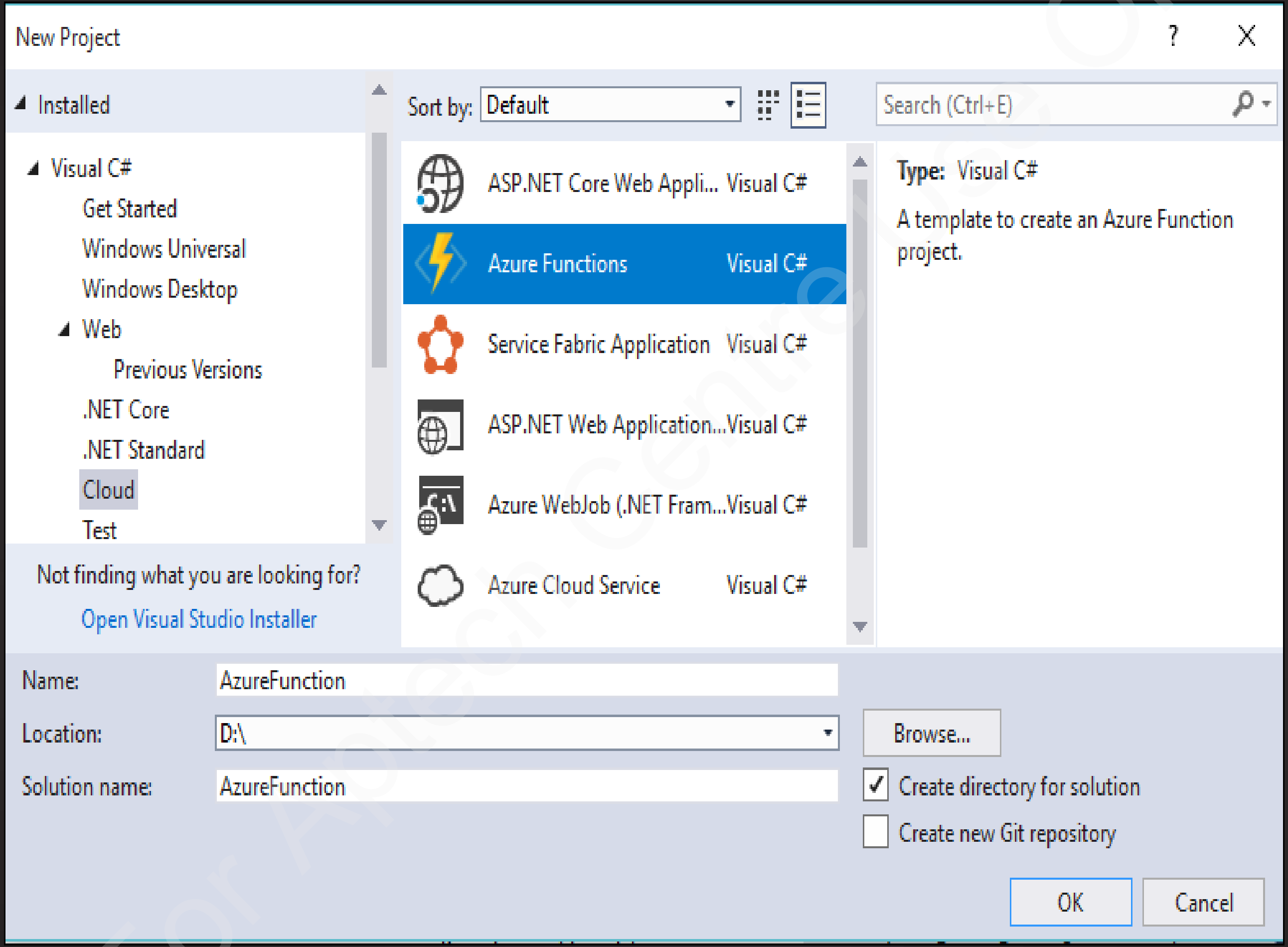
Creating an Assembly [3-7]

```
[assembly: AssemblyTitle("Utility")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("Utility")]
[assembly: AssemblyCopyright("Copyright ©  2018")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
```

Creating an Assembly [4-7]

```
[// Version information for an assembly consists of the following four
values:
//      Major Version
//      Minor Version
//      Build Number
//      Revision
// You can specify all the values or you can default the Build and
Revision Numbers
// by using the '*' as shown:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Creating an Assembly [5-7]


New Project

Installed
- Visual C#
  - Get Started
  - Windows Universal
  - Windows Desktop
  - Web
    - Previous Versions
  - .NET Core
  - .NET Standard
  - Cloud
  - Test

Not finding what you are looking for?
Open Visual Studio Installer

Sort by: Default

Search (Ctrl+E)

ASP.NET Core Web Appli...	Visual C#
Azure Functions	Visual C#
Service Fabric Application	Visual C#
ASP.NET Web Application...	Visual C#
Azure WebJob (.NET Fram...	Visual C#
Azure Cloud Service	Visual C#

Type: Visual C#

A template to create an Azure Function project.

Name: AzureFunction
Location: D:\
Solution name: AzureFunction

Browse...
[✓] Create directory for solution
[ ] Create new Git repository

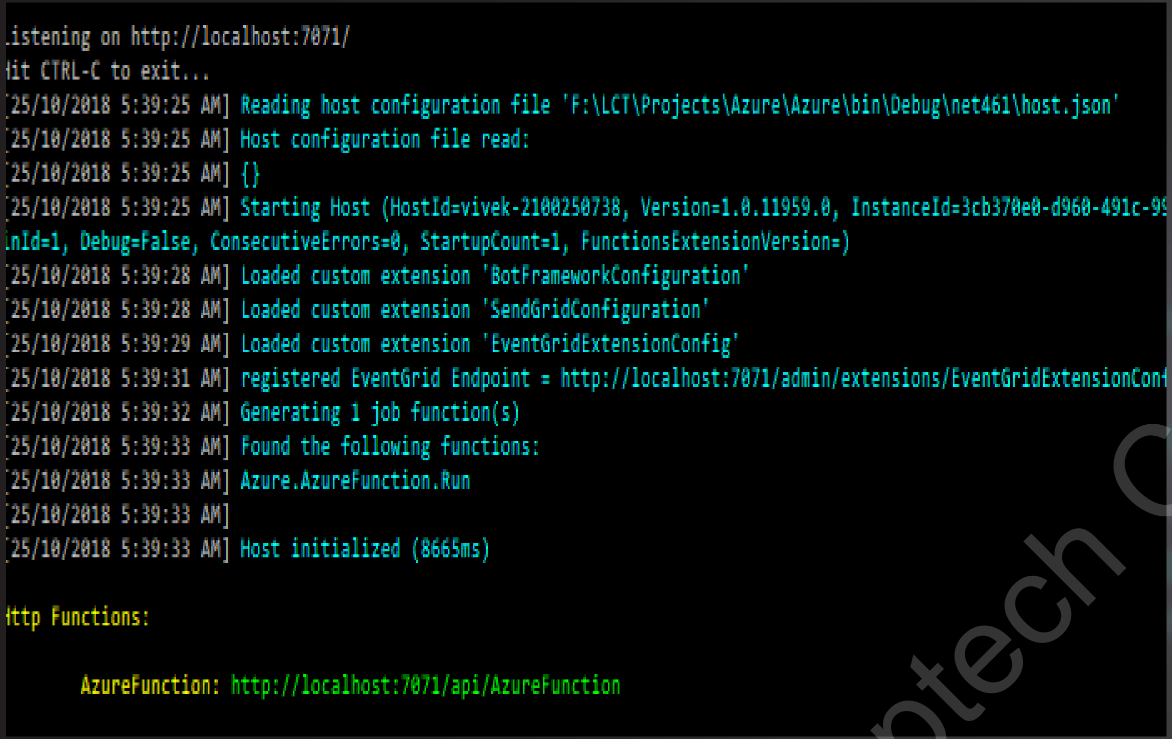
OK Cancel

Creating Azure Function Project

Creating an Assembly [6-7]

```
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Threading.Tasks;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.Azure.WebJobs.Host;
using Utility;
namespace Azure{ public static class AzureFunction {[FunctionName("AzureFunction")]
        public static int Run([HttpTrigger(AuthorizationLevel.Function, "get", "post",
Route = null)]HttpRequestMessage req,TraceWriter log)
        {log.Info("C# HTTP trigger function processed a request.");
            // parse query parameter
         ConvertType obj = new ConvertType();
         int  result= obj.StringToInt("543214");
         return result;} }
}
```

Creating an Assembly [7-7]

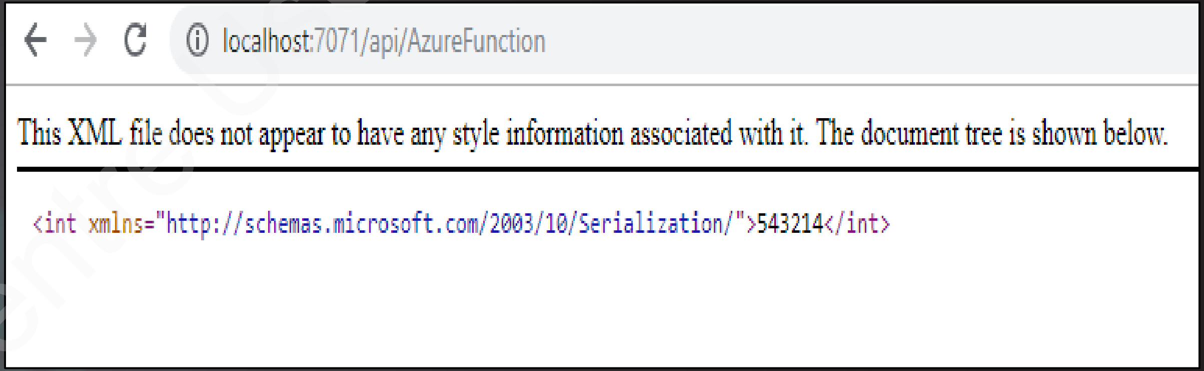


```
.istening on http://localhost:7071/
Hit CTRL-C to exit...
[25/10/2018 5:39:25 AM] Reading host configuration file 'F:\LCT\Projects\Azure\Azure\bin\Debug\net461\host.json'
[25/10/2018 5:39:25 AM] Host configuration file read:
[25/10/2018 5:39:25 AM] {}
[25/10/2018 5:39:25 AM] Starting Host (HostId=vivek-2100250738, Version=1.0.11959.0, InstanceId=3cb370e0-d960-491c-99
.nId=1, Debug=False, ConsecutiveErrors=0, StartupCount=1, FunctionsExtensionVersion=)
[25/10/2018 5:39:28 AM] Loaded custom extension 'BotFrameworkConfiguration'
[25/10/2018 5:39:28 AM] Loaded custom extension 'SendGridConfiguration'
[25/10/2018 5:39:29 AM] Loaded custom extension 'EventGridExtensionConfig'
[25/10/2018 5:39:31 AM] registered EventGrid Endpoint = http://localhost:7071/admin/extensions/EventGridExtensionConf
[25/10/2018 5:39:32 AM] Generating 1 job function(s)
[25/10/2018 5:39:33 AM] Found the following functions:
[25/10/2018 5:39:33 AM] Azure.AzureFunction.Run
[25/10/2018 5:39:33 AM]
[25/10/2018 5:39:33 AM] Host initialized (8665ms)

Http Functions:

        AzureFunction: http://localhost:7071/api/AzureFunction
```

Azure Function URL



localhost:7071/api/AzureFunction

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<int xmlns="http://schemas.microsoft.com/2003/10/Serialization/">543214</int>
```

Implementing Shared Assemblies Across Multiple Servers [1-2]

Assembly Deployment Locations

- In a directory or subdirectories of the application
- On an HTTP server
- In the global assembly cache

Implementing Shared Assemblies Across Multiple Servers [2-2]

Running Assembly in Intranet-Rules for Library Assemblies

Library assemblies should be located in the same folder as the executable assembly.

Assemblies located in a subfolder or that have a reference on another path are not provided with the full-trust grant set.

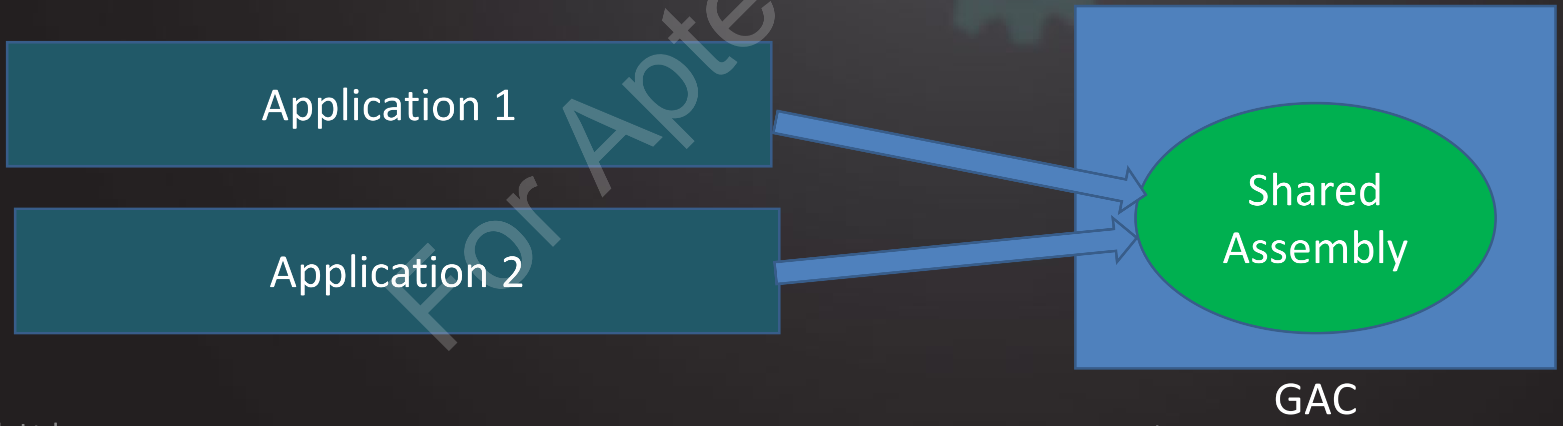
If there is a delay-load of an assembly by the executable, the same path should be used as that for starting the executable.

Former Intranet Policy Restoring

Users could choose to return to the previous action, to provide Intranet evidence by setting a registry key applicable to all applications on the computer.

Deploying Assemblies to the GAC

Command	Details
`gacutil -i <assembly name>`	Shell command to install assembly in Global Assembly Cache
`gacutil -i Utility.dll`	Example installs an assembly of file name `Utility.dll` in the GAC
`gacutil -l` or `gacutil /l`	Shell command to view a list of assemblies in the GAC



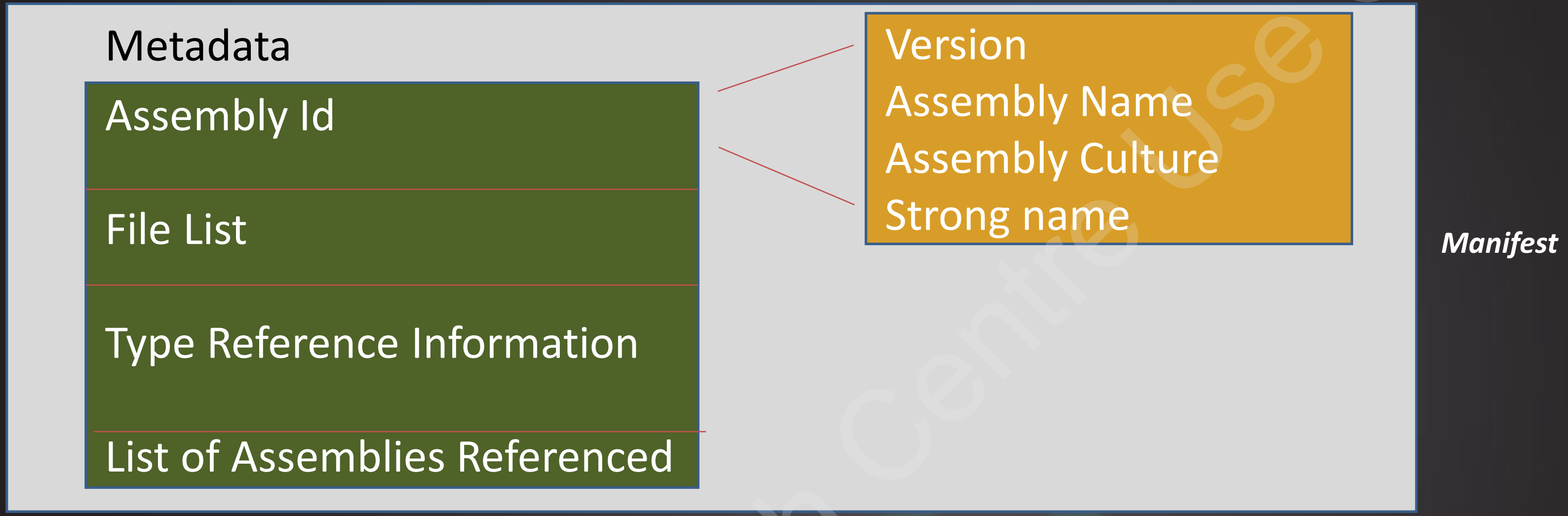
Implementing Assembly Versioning

Major Version	This is incremented manually for major releases.
Minor Version	This is incremented manually for minor releases.
Build Number	This is usually incremented automatically as a part of each build that is carried out on the Build Server.
Revision	This is incremented for QuickiFix ixckis incremQFE), which are also known as hotfixes or patches. This is done to builds that are released in the Production environment (PROD).

```
// by using the '*' as shown:
// [assembly: AssemblyVersion("1.0.*")]
 [assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Creating an Assembly Manifest

Assembly metadata is included in the assembly manifest:



Functions of Assembly Manifest

- Lists files that constitute the assembly.
- Lists other assemblies on which the assembly is dependent.
- Makes the assembly self-describing.

Configuring Assembly Binding Redirects [1-2]

In an application configuration file, `appliesTo` attribute can be used on `<assemblyBinding>` element for redirecting assembly binding references.

Elements of `<assemblyBinding>` are order-sensitive.

```
<runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1"
appliesTo="v2.7.112">
            <dependentAssembly>
               * assembly information goes here *
            </dependentAssembly>
       </assemblyBinding>
</runtime>
```

Configuring Assembly Binding Redirects [2-2]

```
<assemblyBinding xmlns="..." appliesTo="v2.7.112">
<! – .NET Framework version 2.7. redirects here. -->
</assemblyBinding>
<assemblyBinding xmlns="..." appliesTo="v2.8.1">
    <! – .NET Framework version 1.1 redirects here. -->
</assemblyBinding>
<assemblyBinding xmlns="...">
<!-- Redirects meant for all versions of the .NET
Framework. -->
</assemblyBinding>
```

Summary

- An assembly is a single deployable unit containing all the information about implementation of classes, structures, and interfaces.
- Assemblies can be private or shared where, private assembly is accessible by a single application and shared assembly is shared by multiple applications.
- Depending on the implementation, assemblies can be of two types, single-file assemblies or multifile assemblies.
- Assembly versions consist of four different parts Major Version, Minor Version, Build Number, and Revision.