



## CHAPTER 1

System

# *The System Namespace*

CHAPTER SUMMARY GOES HERE

### **Activator**

sealed class

System (mscorlib.dll)

DESCRIPTION OF TYPE GOES HERE

```
public sealed class Activator {  
    // Public Static Methods  
    public static object CreateInstance(Type type);  
    public static object CreateInstance(Type type, System.Reflection.BindingFlags bindingAttr,  
        System.Reflection.Binder binder, object[] args,  
        System.Globalization.CultureInfo culture);  
    public static object CreateInstance(Type type, System.Reflection.BindingFlags bindingAttr,  
        System.Reflection.Binder binder, object[] args,  
        System.Globalization.CultureInfo culture, object[] activationAttributes);  
    public static object CreateInstance(Type type, bool nonPublic);  
    public static object CreateInstance(Type type, object[] args);  
    public static object CreateInstance(Type type, object[] args, object[] activationAttributes);  
    public static ObjectHandle CreateInstance(string assemblyName, string typeName);  
    public static ObjectHandle CreateInstance(string assemblyName, string typeName, bool ignoreCase,  
        System.Reflection.BindingFlags bindingAttr,  
        System.Reflection.Binder binder, object[] args,  
        System.Globalization.CultureInfo culture, object[] activationAttributes,  
        System.Security.Policy.Evidence securityInfo);  
    public static ObjectHandle CreateInstance(string assemblyName, string typeName, object[] activationAttributes);  
    public static ObjectHandle CreateInstanceFrom(string assemblyFile, string typeName);  
    public static ObjectHandle CreateInstanceFrom(string assemblyFile, string typeName, bool ignoreCase,  
        System.Reflection.BindingFlags bindingAttr,  
        System.Reflection.Binder binder, object[] args,  
        System.Globalization.CultureInfo culture,  
        object[] activationAttributes,  
        System.Security.Policy.Evidence securityInfo);  
    public static ObjectHandle CreateInstanceFrom(string assemblyFile, string typeName,  
        object[] activationAttributes);
```

```

public static object GetObject(Type type, string url);
public static object GetObject(Type type, string url, object state);
}

```

## **AppDomain**

**sealed class**

System (mscorlib.dll)

*ECMA, marshal by reference*

DESCRIPTION OF TYPE GOES HERE

```

public sealed class AppDomain : MarshalByRefObject, _AppDomain, System.Security.IEvidenceFactory {
// Public Static Properties
public static AppDomain CurrentDomain {get; }
// Public Instance Properties
public string BaseDirectory {get; } // implements _AppDomain
public string DynamicDirectory {get; } // implements _AppDomain
public Evidence Evidence {get; } // implements System.Security.IEvidenceFactory
public string FriendlyName {get; } // implements _AppDomain
public string RelativeSearchPath {get; } // implements _AppDomain
public AppDomainSetup SetupInformation {get; }
public bool ShadowCopyFiles {get; } // implements _AppDomain
// Public Static Methods
public static AppDomain CreateDomain(string friendlyName);
public static AppDomain CreateDomain(string friendlyName, System.Security.Policy.Evidence securityInfo);
public static AppDomain CreateDomain(string friendlyName, System.Security.Policy.Evidence securityInfo,
AppDomainSetup info);
public static AppDomain CreateDomain(string friendlyName, System.Security.Policy.Evidence securityInfo,
string appBasePath, string appRelativeSearchPath,
bool shadowCopyFiles);
public static int GetCurrentThreadId();
public static void Unload(AppDomain domain);
// Public Instance Methods
public void AppendPrivatePath(string path); // implements _AppDomain
public void ClearPrivatePath(); // implements _AppDomain
public void ClearShadowCopyPath(); // implements _AppDomain
public ObjectHandle CreateInstance(string assemblyName, string typeName); // implements _AppDomain
public ObjectHandle CreateInstance(string assemblyName, string typeName, // implements _AppDomain
bool ignoreCase,
System.Reflection.BindingFlags bindingAttr,
System.Reflection.Binder binder,
object[] args,
System.Globalization.CultureInfo culture,
object[] activationAttributes,
System.Security.Policy.Evidence securityAttributes);
public ObjectHandle CreateInstance(string assemblyName, string typeName, // implements _AppDomain
object[] activationAttributes);
public object CreateInstanceAndUnwrap(string assemblyName, string typeName);
public object CreateInstanceAndUnwrap(string assemblyName, string typeName, bool ignoreCase,
System.Reflection.BindingFlags bindingAttr,
System.Reflection.Binder binder, object[] args,
System.Globalization.CultureInfo culture, object[] activationAttributes,
System.Security.Policy.Evidence securityAttributes);
public object CreateInstanceAndUnwrap(string assemblyName, string typeName, object[] activationAttributes);
public ObjectHandle CreateInstanceFrom(string assemblyFile, string typeName); // implements _AppDomain
public ObjectHandle CreateInstanceFrom(string assemblyFile, string typeName, // implements _AppDomain
bool ignoreCase,
System.Reflection.BindingFlags bindingAttr,
System.Reflection.Binder binder,

```

```

        object[] args,
        System.Globalization.CultureInfo culture,
        object[] activationAttributes,
        System.Security.Policy.Evidence securityAttributes);
public ObjectHandle CreateInstanceFrom(string assemblyFile, string typeName, // implements _AppDomain
        object[] activationAttributes);
public object CreateInstanceFromAndUnwrap(string assemblyName, string typeName);
public object CreateInstanceFromAndUnwrap(string assemblyName, string typeName, bool ignoreCase,
        System.Reflection.BindingFlags bindingAttr,
        System.Reflection.Binder binder, object[] args,
        System.Globalization.CultureInfo culture,
        object[] activationAttributes,
        System.Security.Policy.Evidence securityAttributes);
public object CreateInstanceFromAndUnwrap(string assemblyName, string typeName,
        object[] activationAttributes);
public AssemblyBuilder DefineDynamicAssembly( // implements _AppDomain
        System.Reflection.AssemblyName name,
        System.Reflection.Emit.AssemblyBuilderAccess access);
public AssemblyBuilder DefineDynamicAssembly( // implements _AppDomain
        System.Reflection.AssemblyName name,
        System.Reflection.Emit.AssemblyBuilderAccess access,
        System.Security.Policy.Evidence evidence);
public AssemblyBuilder DefineDynamicAssembly( // implements _AppDomain
        System.Reflection.AssemblyName name,
        System.Reflection.Emit.AssemblyBuilderAccess access,
        System.Security.Policy.Evidence evidence,
        System.Security.PermissionSet requiredPermissions,
        System.Security.PermissionSet optionalPermissions,
        System.Security.PermissionSet refusedPermissions);
public AssemblyBuilder DefineDynamicAssembly( // implements _AppDomain
        System.Reflection.AssemblyName name,
        System.Reflection.Emit.AssemblyBuilderAccess access,
        System.Security.PermissionSet requiredPermissions,
        System.Security.PermissionSet optionalPermissions,
        System.Security.PermissionSet refusedPermissions);
public AssemblyBuilder DefineDynamicAssembly( // implements _AppDomain
        System.Reflection.AssemblyName name,
        System.Reflection.Emit.AssemblyBuilderAccess access,
        string dir);
public AssemblyBuilder DefineDynamicAssembly( // implements _AppDomain
        System.Reflection.AssemblyName name,
        System.Reflection.Emit.AssemblyBuilderAccess access,
        string dir,
        System.Security.Policy.Evidence evidence);
public AssemblyBuilder DefineDynamicAssembly( // implements _AppDomain
        System.Reflection.AssemblyName name,
        System.Reflection.Emit.AssemblyBuilderAccess access,
        string dir,
        System.Security.Policy.Evidence evidence,
        System.Security.PermissionSet requiredPermissions,
        System.Security.PermissionSet optionalPermissions,
        System.Security.PermissionSet refusedPermissions);
public AssemblyBuilder DefineDynamicAssembly( // implements _AppDomain
        System.Reflection.AssemblyName name,
        System.Reflection.Emit.AssemblyBuilderAccess access,
        string dir,
        System.Security.Policy.Evidence evidence,

```

```

        System.Security.PermissionSet requiredPermissions,
        System.Security.PermissionSet optionalPermissions,
        System.Security.PermissionSet refusedPermissions,
        bool isSynchronized);

public AssemblyBuilder DefineDynamicAssembly(                // implements _AppDomain
    System.Reflection.AssemblyName name,
    System.Reflection.Emit.AssemblyBuilderAccess access,
    string dir,
    System.Security.PermissionSet requiredPermissions,
    System.Security.PermissionSet optionalPermissions,
    System.Security.PermissionSet refusedPermissions);

public void DoCallBack(CrossAppDomainDelegate callBackDelegate); // implements _AppDomain
public int ExecuteAssembly(string assemblyFile);                // implements _AppDomain
public int ExecuteAssembly(string assemblyFile,                // implements _AppDomain
    System.Security.Policy.Evidence assemblySecurity);
public int ExecuteAssembly(string assemblyFile,                // implements _AppDomain
    System.Security.Policy.Evidence assemblySecurity,
    string[] args);

public Assembly[] GetAssemblies();                            // implements _AppDomain
public object GetData(string name);                          // implements _AppDomain
public Type GetType();                                        // overrides object
public override object InitializeLifetimeService();          // overrides MarshalByRefObject
public bool IsFinalizingForUnload();

public Assembly Load(System.Reflection.AssemblyName assemblyRef); // implements _AppDomain
public Assembly Load(System.Reflection.AssemblyName assemblyRef, // implements _AppDomain
    System.Security.Policy.Evidence assemblySecurity);
public Assembly Load(byte[] rawAssembly);                    // implements _AppDomain
public Assembly Load(byte[] rawAssembly, byte[] rawSymbolStore); // implements _AppDomain
public Assembly Load(byte[] rawAssembly, byte[] rawSymbolStore, // implements _AppDomain
    System.Security.Policy.Evidence securityEvidence);
public Assembly Load(string assemblyString);                 // implements _AppDomain
public Assembly Load(string assemblyString,                 // implements _AppDomain
    System.Security.Policy.Evidence assemblySecurity);
public void SetAppDomainPolicy(System.Security.Policy.PolicyLevel domainPolicy); // implements _AppDomain
public void SetCachePath(string path);                      // implements _AppDomain
public void SetData(string name, object data);             // implements _AppDomain
public void SetDynamicBase(string path);
public void SetPrincipalPolicy(System.Security.Principal.PrincipalPolicy policy); // implements _AppDomain
public void SetShadowCopyFiles();
public void SetShadowCopyPath(string path);                // implements _AppDomain
public void SetThreadPrincipal(System.Security.Principal.IPrincipal principal); // implements _AppDomain
public override string Tostring();                           // overrides object
// Events
public event AssemblyLoadEventHandler AssemblyLoad;         // implements _AppDomain
public event ResolveEventHandler AssemblyResolve;           // implements _AppDomain
public event EventHandler DomainUnload;                    // implements _AppDomain
public event EventHandler ProcessExit;                      // implements _AppDomain
public event ResolveEventHandler ResourceResolve;           // implements _AppDomain
public event ResolveEventHandler TypeResolve;               // implements _AppDomain
public event UnhandledExceptionEventHandler UnhandledException; // implements _AppDomain
}

```

*Hierarchy*: Object → MarshalByRefObject → AppDomain(\_AppDomain, System.Security.IEvidenceFactory)

*Returned By*: System.Threading.Thread.GetDomain()

## AppDomainSetup

sealed class

---

**System (mscorlib.dll)** *serializable*

DESCRIPTION OF TYPE GOES HERE

```
public sealed class AppDomainSetup : IAppDomainSetup {
    // Public Constructors
    public AppDomainSetup();
    // Public Instance Properties
    public string ApplicationBase {set; get;} // implements IAppDomainSetup
    public string ApplicationName {set; get;} // implements IAppDomainSetup
    public string CachePath {set; get;} // implements IAppDomainSetup
    public string ConfigurationFile {set; get;} // implements IAppDomainSetup
    public bool DisallowPublisherPolicy {set; get;}
    public string DynamicBase {set; get;} // implements IAppDomainSetup
    public string LicenseFile {set; get;} // implements IAppDomainSetup
    public LoaderOptimization LoaderOptimization {set; get;}
    public string PrivateBinPath {set; get;} // implements IAppDomainSetup
    public string PrivateBinPathProbe {set; get;} // implements IAppDomainSetup
    public string ShadowCopyDirectories {set; get;} // implements IAppDomainSetup
    public string ShadowCopyFiles {set; get;} // implements IAppDomainSetup
}
```

*Returned By:* AppDomain.SetupInformation*Passed To:* AppDomain.CreateDomain()

---

**AppDomainUnloadedException** **class**
**System (mscorlib.dll)** *serializable*

DESCRIPTION OF TYPE GOES HERE

```
public class AppDomainUnloadedException : SystemException {
    // Public Constructors
    public AppDomainUnloadedException();
    public AppDomainUnloadedException(string message);
    public AppDomainUnloadedException(string message, Exception innerException);
    // Protected Constructors
    protected AppDomainUnloadedException(System.Runtime.Serialization.SerializationInfo info,
        System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.IEnumerable) → SystemException → AppDomainUnloadedException

---

**ApplicationException** **class**
**System (mscorlib.dll)** *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public class ApplicationException : Exception {
    // Public Constructors
    public ApplicationException();
    public ApplicationException(string message);
    public ApplicationException(string message, Exception innerException);
    // Protected Constructors
    protected ApplicationException(System.Runtime.Serialization.SerializationInfo info,
        System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → ApplicationException

*Subclasses:* System.Reflection.InvalidFilterCriteriaException, TargetException, TargetInvocationException, TargetParameterCountException

## **ArgIterator** struct

System (mscorlib.dll)

DESCRIPTION OF TYPE GOES HERE

```
public struct ArgIterator {  
    // Public Constructors  
    public ArgIterator(RuntimeArgumentHandle arglist);  
    public ArgIterator(RuntimeArgumentHandle arglist, void *ptr);  
    // Public Instance Methods  
    public void End();  
    public override bool Equals(object o); // overrides ValueType  
    public override int GetHashCode(); // overrides ValueType  
    public TypedReference GetNextArg();  
    public TypedReference GetNextArg(RuntimeTypeHandle rth);  
    public RuntimeTypeHandle GetNextArgType();  
    public int GetRemainingCount();  
}
```

*Hierarchy:* Object → ValueType → ArgIterator

## **ArgumentException** class

System (mscorlib.dll) ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class ArgumentException : SystemException {  
    // Public Constructors  
    public ArgumentException();  
    public ArgumentException(string message);  
    public ArgumentException(string message, Exception innerException);  
    public ArgumentException(string message, string paramName);  
    public ArgumentException(string message, string paramName, Exception innerException);  
    // Protected Constructors  
    protected ArgumentException(System.Runtime.Serialization.SerializationInfo info,  
        System.Runtime.Serialization.StreamingContext context);  
    // Public Instance Properties  
    public override string Message {get;} // overrides Exception  
    public virtual string ParamName {get;}  
    // Public Instance Methods  
    public override void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // overrides Exception  
        System.Runtime.Serialization.StreamingContext context);  
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → ArgumentException

*Subclasses:* ArgumentNullException, ArgumentOutOfRangeException, DuplicateWaitObjectException

## **ArgumentNullException** class

System (mscorlib.dll) ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class ArgumentNullException : ArgumentException {
// Public Constructors
public ArgumentNullException();
public ArgumentNullException(string paramName);
public ArgumentNullException(string paramName, string message);
// Protected Constructors
protected ArgumentNullException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → ArgumentException → ArgumentNullException

## **ArgumentOutOfRangeException**

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class ArgumentOutOfRangeException : ArgumentException {
// Public Constructors
public ArgumentOutOfRangeException();
public ArgumentOutOfRangeException(string paramName);
public ArgumentOutOfRangeException(string paramName, object actualValue, string message);
public ArgumentOutOfRangeException(string paramName, string message);
// Protected Constructors
protected ArgumentOutOfRangeException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
// Public Instance Properties
public virtual object ActualValue {get; }
public override string Message {get; } // overrides ArgumentException
// Public Instance Methods
public override void GetObjectData( // overrides ArgumentException
System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → ArgumentException → ArgumentOutOfRangeException

## **ArithmeticException**

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class ArithmeticException : SystemException {
// Public Constructors
public ArithmeticException();
public ArithmeticException(string message);
public ArithmeticException(string message, Exception innerException);
// Protected Constructors
protected ArithmeticException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → ArithmeticException

*Subclasses:* DivideByZeroException, NotFiniteNumberException, OverflowException

## Array

**abstract class**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public abstract class Array : ICloneable, IList, ICollection, IEnumerable {  
    // Public Instance Properties  
    public virtual bool IsFixedSize {get; } // implements IList  
    public virtual bool IsReadOnly {get; } // implements IList  
    public virtual bool IsSynchronized {get; } // implements ICollection  
    public int Length {get; }  
    public int Rank {get; }  
    public virtual object SyncRoot {get; } // implements ICollection  
    // Public Static Methods  
    public static int BinarySearch(Array array, int index, int length, object value);  
    public static int BinarySearch(Array array, int index, int length, object value,  
        System.Collections.IComparer comparer);  
    public static int BinarySearch(Array array, object value);  
    public static int BinarySearch(Array array, object value, System.Collections.IComparer comparer);  
    public static void Clear(Array array, int index, int length);  
    public static void Copy(Array sourceArray, Array destinationArray, int length);  
    public static void Copy(Array sourceArray, int sourceIndex, Array destinationArray, int destinationIndex, int length);  
    public static Array CreateInstance(Type elementType, int length);  
    public static Array CreateInstance(Type elementType, int[] lengths);  
    public static Array CreateInstance(Type elementType, int[] lengths, int[] lowerBounds);  
    public static Array CreateInstance(Type elementType, int length1, int length2);  
    public static Array CreateInstance(Type elementType, int length1, int length2, int length3);  
    public static int IndexOf(Array array, object value);  
    public static int IndexOf(Array array, object value, int startIndex);  
    public static int IndexOf(Array array, object value, int startIndex, int count);  
    public static int LastIndexOf(Array array, object value);  
    public static int LastIndexOf(Array array, object value, int startIndex);  
    public static int LastIndexOf(Array array, object value, int startIndex, int count);  
    public static void Reverse(Array array);  
    public static void Reverse(Array array, int index, int length);  
    public static void Sort(Array array);  
    public static void Sort(Array keys, Array items);  
    public static void Sort(Array keys, Array items, System.Collections.IComparer comparer);  
    public static void Sort(Array keys, Array items, int index, int length);  
    public static void Sort(Array keys, Array items, int index, int length, System.Collections.IComparer comparer);  
    public static void Sort(Array array, System.Collections.IComparer comparer);  
    public static void Sort(Array array, int index, int length);  
    public static void Sort(Array array, int index, int length, System.Collections.IComparer comparer);  
    // Public Instance Methods  
    public virtual object Clone(); // implements ICloneable  
    public virtual void CopyTo(Array array, int index); // implements ICollection  
    public virtual IEnumerator GetEnumerator(); // implements IEnumerable  
    public int GetLength(int dimension);  
    public int GetLowerBound(int dimension);  
    public int GetUpperBound(int dimension);  
    public object GetValue(int index);  
    public object GetValue(int[] indices);  
    public object GetValue(int index1, int index2);  
    public object GetValue(int index1, int index2, int index3);  
    public void Initialize();
```

```

public void SetValue(object value, int index);
public void SetValue(object value, int[] indices);
public void SetValue(object value, int index1, int index2);
public void SetValue(object value, int index1, int index2, int index3);
}

```

*Returned By:* System.Collections.ArrayList.ToArray(), Enum.GetValues()

*Passed To:* Multiple types

## ArrayTypeMismatchException

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class ArrayTypeMismatchException : SystemException {
// Public Constructors
public ArrayTypeMismatchException();
public ArrayTypeMismatchException(string message);
public ArrayTypeMismatchException(string message, Exception innerException);
// Protected Constructors
protected ArrayTypeMismatchException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → ArrayTypeMismatchException

## AssemblyLoadEventArgs

class

System (mscorlib.dll)

ECMA

DESCRIPTION OF TYPE GOES HERE

```

public class AssemblyLoadEventArgs : EventArgs {
// Public Constructors
public AssemblyLoadEventArgs(System.Reflection.Assembly loadedAssembly);
// Public Instance Properties
public Assembly LoadedAssembly {get; }
}

```

*Hierarchy:* Object → EventArgs → AssemblyLoadEventArgs

*Passed To:* AssemblyLoadEventHandler.{BeginInvoke(), Invoke()}

## AssemblyLoadEventHandler

delegate

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public delegate void AssemblyLoadEventHandler(object sender, AssemblyLoadEventArgs args);

```

*Associated Events:* AppDomain.AssemblyLoad()

## AsyncCallback

delegate

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public delegate void AsyncCallback(IAsyncResult ar);
```

*Passed To:* Multiple types

## **Attribute**

**abstract class**

**System (mscorlib.dll)**

*ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public abstract class Attribute {  
    // Protected Constructors  
    protected Attribute();  
    // Public Instance Properties  
    public virtual object TypeId {get; }  
    // Public Static Methods  
    public static Attribute GetCustomAttribute(System.Reflection.Assembly element, Type attributeType);  
    public static Attribute GetCustomAttribute(System.Reflection.Assembly element, Type attributeType, bool inherit);  
    public static Attribute GetCustomAttribute(System.Reflection.MemberInfo element, Type attributeType);  
    public static Attribute GetCustomAttribute(System.Reflection.MemberInfo element, Type attributeType,  
        bool inherit);  
    public static Attribute GetCustomAttribute(System.Reflection.Module element, Type attributeType);  
    public static Attribute GetCustomAttribute(System.Reflection.Module element, Type attributeType, bool inherit);  
    public static Attribute GetCustomAttribute(System.Reflection.ParameterInfo element, Type attributeType);  
    public static Attribute GetCustomAttribute(System.Reflection.ParameterInfo element, Type attributeType,  
        bool inherit);  
    public static Attribute[] GetCustomAttributes(System.Reflection.Assembly element);  
    public static Attribute[] GetCustomAttributes(System.Reflection.Assembly element, bool inherit);  
    public static Attribute[] GetCustomAttributes(System.Reflection.Assembly element, Type attributeType);  
    public static Attribute[] GetCustomAttributes(System.Reflection.Assembly element, Type attributeType,  
        bool inherit);  
    public static Attribute[] GetCustomAttributes(System.Reflection.MemberInfo element);  
    public static Attribute[] GetCustomAttributes(System.Reflection.MemberInfo element, bool inherit);  
    public static Attribute[] GetCustomAttributes(System.Reflection.MemberInfo element, Type type);  
    public static Attribute[] GetCustomAttributes(System.Reflection.MemberInfo element, Type type, bool inherit);  
    public static Attribute[] GetCustomAttributes(System.Reflection.Module element);  
    public static Attribute[] GetCustomAttributes(System.Reflection.Module element, bool inherit);  
    public static Attribute[] GetCustomAttributes(System.Reflection.Module element, Type attributeType);  
    public static Attribute[] GetCustomAttributes(System.Reflection.Module element, Type attributeType,  
        bool inherit);  
    public static Attribute[] GetCustomAttributes(System.Reflection.ParameterInfo element);  
    public static Attribute[] GetCustomAttributes(System.Reflection.ParameterInfo element, bool inherit);  
    public static Attribute[] GetCustomAttributes(System.Reflection.ParameterInfo element, Type attributeType);  
    public static Attribute[] GetCustomAttributes(System.Reflection.ParameterInfo element, Type attributeType,  
        bool inherit);  
    public static bool IsDefined(System.Reflection.Assembly element, Type attributeType);  
    public static bool IsDefined(System.Reflection.Assembly element, Type attributeType, bool inherit);  
    public static bool IsDefined(System.Reflection.MemberInfo element, Type attributeType);  
    public static bool IsDefined(System.Reflection.MemberInfo element, Type attributeType, bool inherit);  
    public static bool IsDefined(System.Reflection.Module element, Type attributeType);  
    public static bool IsDefined(System.Reflection.Module element, Type attributeType, bool inherit);  
    public static bool IsDefined(System.Reflection.ParameterInfo element, Type attributeType);  
    public static bool IsDefined(System.Reflection.ParameterInfo element, Type attributeType, bool inherit);  
    // Public Instance Methods  
    public override bool Equals(object obj); // overrides object  
    public override int GetHashCode(); // overrides object
```

```

public virtual bool IsDefaultAttribute();
public virtual bool Match(object obj);
}

```

*Subclasses:* Multiple types

*Valid On:* All

## AttributeTargets

enum

System (mscorlib.dll)

ECMA, serializable, flag

DESCRIPTION OF TYPE GOES HERE

```

public enum AttributeTargets {
    Assembly = 0x00000001,
    Module = 0x00000002,
    Class = 0x00000004,
    Struct = 0x00000008,
    Enum = 0x00000010,
    Constructor = 0x00000020,
    Method = 0x00000040,
    Property = 0x00000080,
    Field = 0x00000100,
    Event = 0x00000200,
    Interface = 0x00000400,
    Parameter = 0x00000800,
    Delegate = 0x00001000,
    ReturnValue = 0x00002000,
    All = 0x00003FFF
}

```

*Hierarchy:* Object → ValueType → Enum(IComparable, IFormattable, IConvertible) → AttributeTargets

*Returned By:* AttributeUsageAttribute.ValidOn

*Passed To:* AttributeUsageAttribute.AttributeUsageAttribute()

## AttributeUsageAttribute

sealed class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public sealed class AttributeUsageAttribute : Attribute {
    // Public Constructors
    public AttributeUsageAttribute(AttributeTargets validOn);
    // Public Instance Properties
    public bool AllowMultiple {set; get; }
    public bool Inherited {set; get; }
    public AttributeTargets ValidOn {get; }
}

```

*Hierarchy:* Object → Attribute → AttributeUsageAttribute

*Valid On:* Class

## BadImageFormatException

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class BadImageFormatException : SystemException {  
    // Public Constructors  
    public BadImageFormatException();  
    public BadImageFormatException(string message);  
    public BadImageFormatException(string message, Exception inner);  
    public BadImageFormatException(string message, string fileName);  
    public BadImageFormatException(string message, string fileName, Exception inner);  
    // Protected Constructors  
    protected BadImageFormatException(System.Runtime.Serialization.SerializationInfo info,  
                                        System.Runtime.Serialization.StreamingContext context);  
    // Public Instance Properties  
    public string FileName {get; }  
    public string FusionLog {get; }  
    public override string Message {get; } // overrides Exception  
    // Public Instance Methods  
    public override void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // overrides Exception  
                                        System.Runtime.Serialization.StreamingContext context);  
    public override string ToString(); // overrides Exception  
}
```

*Hierarchy*: Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → BadImageFormatException

## BitConverter

sealed class

System (mscorlib.dll)

DESCRIPTION OF TYPE GOES HERE

```
public sealed class BitConverter {  
    // Public Static Fields  
    public static readonly bool IsLittleEndian; //False  
    // Public Static Methods  
    public static long DoubleToInt64Bits(double value);  
    public static byte[] GetBytes(bool value);  
    public static byte[] GetBytes(char value);  
    public static byte[] GetBytes(double value);  
    public static byte[] GetBytes(short value);  
    public static byte[] GetBytes(int value);  
    public static byte[] GetBytes(long value);  
    public static byte[] GetBytes(float value);  
    public static byte[] GetBytes(ushort value);  
    public static byte[] GetBytes(uint value);  
    public static byte[] GetBytes(ulong value);  
    public static double Int64BitsToDouble(long value);  
    public static bool ToBoolean(byte[] value, int startIndex);  
    public static char ToChar(byte[] value, int startIndex);  
    public static double ToDouble(byte[] value, int startIndex);  
    public static short ToInt16(byte[] value, int startIndex);  
    public static int ToInt32(byte[] value, int startIndex);  
    public static long ToInt64(byte[] value, int startIndex);  
    public static float ToSingle(byte[] value, int startIndex);  
    public static string ToString(byte[] value);  
    public static string ToString(byte[] value, int startIndex);  
    public static string ToString(byte[] value, int startIndex, int length);  
    public static ushort ToUInt16(byte[] value, int startIndex);  
    public static uint ToUInt32(byte[] value, int startIndex);  
}
```

```
public static ulong ToUInt64(byte[] value, int startIndex);
}
```

**Boolean****struct**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct Boolean : IComparable, IConvertible {
// Public Static Fields
public static readonly string FalseString;           =False
public static readonly string TrueString;           =True
// Public Static Methods
public static bool Parse(string value);
// Public Instance Methods
public int CompareTo(object obj);                    // implements IComparable
public override bool Equals(object obj);             // overrides ValueType
public override int GetHashCode();                  // overrides ValueType
public TypeCode GetTypeCode();                     // implements IConvertible
public override string ToString();                  // overrides ValueType
public string ToString(IFormatProvider provider);    // implements IConvertible
}
```

*Hierarchy:* Object → ValueType → Boolean(IComparable, IConvertible)*Returned By:* Multiple types*Passed To:* Multiple types**Buffer****sealed class**

System (mscorlib.dll)

DESCRIPTION OF TYPE GOES HERE

```
public sealed class Buffer {
// Public Static Methods
public static void BlockCopy(Array src, int srcOffset, Array dst, int dstOffset, int count);
public static int ByteLength(Array array);
public static byte GetByte(Array array, int index);
public static void SetByte(Array array, int index, byte value);
}
```

**Byte****struct**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct Byte : IComparable, IFormattable, IConvertible {
// Public Static Fields
public const byte MaxValue;                         =255
public const byte MinValue;                         =0
// Public Static Methods
public static byte Parse(string s);
public static byte Parse(string s, IFormatProvider provider);
public static byte Parse(string s, System.Globalization.NumberStyles style);
public static byte Parse(string s, System.Globalization.NumberStyles style, IFormatProvider provider);
// Public Instance Methods
public int CompareTo(object value);                  // implements IComparable
}
```

```

public override bool Equals(object obj); // overrides ValueType
public override int GetHashCode(); // overrides ValueType
public TypeCode GetTypeCode(); // implements IConvertible
public override string ToString(); // overrides ValueType
public string ToString(IFormatProvider provider); // implements IConvertible
public string ToString(string format);
public string ToString(string format, IFormatProvider provider); // implements IFormattable
}

```

*Hierarchy:* Object → ValueType → Byte(IComparable, IFormattable, IConvertible)

*Returned By:* Multiple types

*Passed To:* Multiple types

## **CannotUnloadAppDomainException**

**class**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class CannotUnloadAppDomainException : SystemException {
// Public Constructors
public CannotUnloadAppDomainException();
public CannotUnloadAppDomainException(string message);
public CannotUnloadAppDomainException(string message, Exception innerException);
// Protected Constructors
protected CannotUnloadAppDomainException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → CannotUnloadAppDomainException

## **Char**

**struct**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public struct Char : IComparable, IConvertible {
// Public Static Fields
public const char MaxValue; // =0x0000FFFF
public const char MinValue; // =0x00000000
// Public Static Methods
public static double GetNumericValue(char c);
public static double GetNumericValue(string s, int index);
public static UnicodeCategory GetUnicodeCategory(char c);
public static UnicodeCategory GetUnicodeCategory(string s, int index);
public static bool IsControl(char c);
public static bool IsControl(string s, int index);
public static bool IsDigit(char c);
public static bool IsDigit(string s, int index);
public static bool IsLetter(char c);
public static bool IsLetter(string s, int index);
public static bool IsLetterOrDigit(char c);
public static bool IsLetterOrDigit(string s, int index);
public static bool IsLower(char c);
public static bool IsLower(string s, int index);
public static bool IsNumber(char c);
}

```

```

public static bool IsNumber(string s, int index);
public static bool IsPunctuation(char c);
public static bool IsPunctuation(string s, int index);
public static bool IsSeparator(char c);
public static bool IsSeparator(string s, int index);
public static bool IsSurrogate(char c);
public static bool IsSurrogate(string s, int index);
public static bool IsSymbol(char c);
public static bool IsSymbol(string s, int index);
public static bool IsUpper(char c);
public static bool IsUpper(string s, int index);
public static bool IsWhiteSpace(char c);
public static bool IsWhiteSpace(string s, int index);
public static char Parse(string s);
public static char ToLower(char c);
public static char ToLower(char c, System.Globalization.CultureInfo culture);
public static string ToString(char c);
public static char ToUpper(char c);
public static char ToUpper(char c, System.Globalization.CultureInfo culture);
// Public Instance Methods
public int CompareTo(object value); // implements IComparable
public override bool Equals(object obj); // overrides ValueType
public override int GetHashCode(); // overrides ValueType
public TypeCode GetTypeCode(); // implements IConvertible
public override string ToString(); // overrides ValueType
public string Tostring(IFormatProvider provider); // implements IConvertible
}

```

*Hierarchy:* Object → ValueType → Char(IComparable, IConvertible)

*Returned By:* Multiple types

*Passed To:* Multiple types

**CharEnumerator** sealed class

System (mscorlib.dll) ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public sealed class CharEnumerator : IEnumerator, ICloneable {
// Public Instance Properties
public char Current {get; }
// Public Instance Methods
public object Clone(); // implements ICloneable
public bool MoveNext(); // implements IEnumerator
public void Reset(); // implements IEnumerator
}

```

*Returned By:* String.GetEnumerator()

**CLSCompliantAttribute** sealed class

System (mscorlib.dll) ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public sealed class CLSCompliantAttribute : Attribute {
// Public Constructors
public CLSCompliantAttribute(bool isCompliant);
}

```

```
// Public Instance Properties
public bool IsCompliant {get; }
}
```

*Hierarchy:* Object → Attribute → CLSCompliantAttribute

*Valid On:* All

## Console

sealed class

System (mscorlib.dll)

ECMA

DESCRIPTION OF TYPE GOES HERE

```
public sealed class Console {
// Public Static Properties
public static TextWriter Error {get; }
public static TextReader In {get; }
public static TextWriter Out {get; }
// Public Static Methods
public static Stream OpenStandardError();
public static Stream OpenStandardError(int bufferSize);
public static Stream OpenStandardInput();
public static Stream OpenStandardInput(int bufferSize);
public static Stream OpenStandardOutput();
public static Stream OpenStandardOutput(int bufferSize);
public static int Read();
public static string ReadLine();
public static void SetError(System.IO.TextWriter newError);
public static void SetIn(System.IO.TextReader newIn);
public static void SetOut(System.IO.TextWriter newOut);
public static void Write(bool value);
public static void Write(char value);
public static void Write(char[] buffer);
public static void Write(char[] buffer, int index, int count);
public static void Write(decimal value);
public static void Write(double value);
public static void Write(int value);
public static void Write(long value);
public static void Write(object value);
public static void Write(float value);
public static void Write(string value);
public static void Write(string format, object arg0);
public static void Write(string format, params object[] arg);
public static void Write(string format, object arg0, object arg1);
public static void Write(string format, object arg0, object arg1, object arg2);
public static void Write(string format, object arg0, object arg1, object arg2, object arg3);
public static void Write(uint value);
public static void Write(ulong value);
public static void WriteLine();
public static void WriteLine(bool value);
public static void WriteLine(char value);
public static void WriteLine(char[] buffer);
public static void WriteLine(char[] buffer, int index, int count);
public static void WriteLine(decimal value);
public static void WriteLine(double value);
public static void WriteLine(int value);
public static void WriteLine(long value);
```

```

public static void WriteLine(object value);
public static void WriteLine(float value);
public static void WriteLine(string value);
public static void WriteLine(string format, object arg0);
public static void WriteLine(string format, params object[] arg);
public static void WriteLine(string format, object arg0, object arg1);
public static void WriteLine(string format, object arg0, object arg1, object arg2);
public static void WriteLine(string format, object arg0, object arg1, object arg2, object arg3);
public static void WriteLine(uint value);
public static void WriteLine(ulong value);
}

```

### ContextBoundObject abstract class

System (mscorlib.dll) *serializable, marshal by reference, context bound*

DESCRIPTION OF TYPE GOES HERE

```

public abstract class ContextBoundObject : MarshalByRefObject {
// Protected Constructors
protected ContextBoundObject();
}

```

*Hierarchy:* Object → MarshalByRefObject → ContextBoundObject

### ContextMarshalException class

System (mscorlib.dll) *serializable*

DESCRIPTION OF TYPE GOES HERE

```

public class ContextMarshalException : SystemException {
// Public Constructors
public ContextMarshalException();
public ContextMarshalException(string message);
public ContextMarshalException(string message, Exception inner);
// Protected Constructors
protected ContextMarshalException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → ContextMarshalException

### ContextStaticAttribute class

System (mscorlib.dll) *serializable*

DESCRIPTION OF TYPE GOES HERE

```

public class ContextStaticAttribute : Attribute {
// Public Constructors
public ContextStaticAttribute();
}

```

*Hierarchy:* Object → Attribute → ContextStaticAttribute

*Valid On:* Field

### Convert sealed class

System (mscorlib.dll) *ECMA*

DESCRIPTION OF TYPE GOES HERE

```
public sealed class Convert {  
    // Public Static Fields  
    public static readonly object DBNull;  
    // Public Static Methods  
    public static object ChangeType(object value, Type conversionType);  
    public static object ChangeType(object value, TypeCode typeCode);  
    public static object ChangeType(object value, TypeCode typeCode, IFormatProvider provider);  
    public static object ChangeType(object value, Type conversionType, IFormatProvider provider);  
    public static byte[] FromBase64CharArray(char[] inArray, int offset, int length);  
    public static byte[] FromBase64String(string s);  
    public static TypeCode GetTypeCode(object value);  
    public static bool IsDBNull(object value);  
    public static int ToBase64CharArray(byte[] inArray, int offsetIn, int length, char[] outArray, int offsetOut);  
    public static string ToBase64String(byte[] inArray);  
    public static string ToBase64String(byte[] inArray, int offset, int length);  
    public static bool ToBoolean(bool value);  
    public static bool ToBoolean(byte value);  
    public static bool ToBoolean(char value);  
    public static bool ToBoolean(DateTime value);  
    public static bool ToBoolean(decimal value);  
    public static bool ToBoolean(double value);  
    public static bool ToBoolean(short value);  
    public static bool ToBoolean(int value);  
    public static bool ToBoolean(long value);  
    public static bool ToBoolean(object value);  
    public static bool ToBoolean(object value, IFormatProvider provider);  
    public static bool ToBoolean(sbyte value);  
    public static bool ToBoolean(float value);  
    public static bool ToBoolean(string value);  
    public static bool ToBoolean(string value, IFormatProvider provider);  
    public static bool ToBoolean(ushort value);  
    public static bool ToBoolean(uint value);  
    public static bool ToBoolean(ulong value);  
    public static byte ToByte(bool value);  
    public static byte ToByte(byte value);  
    public static byte ToByte(char value);  
    public static byte ToByte(DateTime value);  
    public static byte ToByte(decimal value);  
    public static byte ToByte(double value);  
    public static byte ToByte(short value);  
    public static byte ToByte(int value);  
    public static byte ToByte(long value);  
    public static byte ToByte(object value);  
    public static byte ToByte(object value, IFormatProvider provider);  
    public static byte ToByte(sbyte value);  
    public static byte ToByte(float value);  
    public static byte ToByte(string value);  
    public static byte ToByte(string value, IFormatProvider provider);  
    public static byte ToByte(string value, int fromBase);  
    public static byte ToByte(ushort value);  
    public static byte ToByte(uint value);  
    public static byte ToByte(ulong value);  
    public static char ToChar(bool value);  
    public static char ToChar(byte value);  
    public static char ToChar(char value);
```

```
public static char ToChar(DateTime value);
public static char ToChar(decimal value);
public static char ToChar(double value);
public static char ToChar(short value);
public static char ToChar(int value);
public static char ToChar(long value);
public static char ToChar(object value);
public static char ToChar(object value, IFormatProvider provider);
public static char ToChar(sbyte value);
public static char ToChar(float value);
public static char ToChar(string value);
public static char ToChar(string value, IFormatProvider provider);
public static char ToChar(ushort value);
public static char ToChar(uint value);
public static char ToChar(ulong value);
public static DateTime ToDateTime(bool value);
public static DateTime ToDateTime(byte value);
public static DateTime ToDateTime(char value);
public static DateTime ToDateTime(DateTime value);
public static DateTime ToDateTime(decimal value);
public static DateTime ToDateTime(double value);
public static DateTime ToDateTime(short value);
public static DateTime ToDateTime(int value);
public static DateTime ToDateTime(long value);
public static DateTime ToDateTime(object value);
public static DateTime ToDateTime(object value, IFormatProvider provider);
public static DateTime ToDateTime(sbyte value);
public static DateTime ToDateTime(float value);
public static DateTime ToDateTime(string value);
public static DateTime ToDateTime(string value, IFormatProvider provider);
public static DateTime ToDateTime(ushort value);
public static DateTime ToDateTime(uint value);
public static DateTime ToDateTime(ulong value);
public static decimal ToDecimal(bool value);
public static decimal ToDecimal(byte value);
public static decimal ToDecimal(char value);
public static decimal ToDecimal(DateTime value);
public static decimal ToDecimal(decimal value);
public static decimal ToDecimal(double value);
public static decimal ToDecimal(short value);
public static decimal ToDecimal(int value);
public static decimal ToDecimal(long value);
public static decimal ToDecimal(object value);
public static decimal ToDecimal(object value, IFormatProvider provider);
public static decimal ToDecimal(sbyte value);
public static decimal ToDecimal(float value);
public static decimal ToDecimal(string value);
public static decimal ToDecimal(string value, IFormatProvider provider);
public static decimal ToDecimal(ushort value);
public static decimal ToDecimal(uint value);
public static decimal ToDecimal(ulong value);
public static double ToDouble(bool value);
public static double ToDouble(byte value);
public static double ToDouble(char value);
public static double ToDouble(DateTime value);
```

```

public static double ToDouble(decimal value);
public static double ToDouble(double value);
public static double ToDouble(short value);
public static double ToDouble(int value);
public static double ToDouble(long value);
public static double ToDouble(object value);
public static double ToDouble(object value, IFormatProvider provider);
public static double ToDouble(sbyte value);
public static double ToDouble(float value);
public static double ToDouble(string value);
public static double ToDouble(string value, IFormatProvider provider);
public static double ToDouble(ushort value);
public static double ToDouble(uint value);
public static double ToDouble(ulong value);
public static short ToInt16(bool value);
public static short ToInt16(byte value);
public static short ToInt16(char value);
public static short ToInt16(DateTime value);
public static short ToInt16(decimal value);
public static short ToInt16(double value);
public static short ToInt16(short value);
public static short ToInt16(int value);
public static short ToInt16(long value);
public static short ToInt16(object value);
public static short ToInt16(object value, IFormatProvider provider);
public static short ToInt16(sbyte value);
public static short ToInt16(float value);
public static short ToInt16(string value);
public static short ToInt16(string value, IFormatProvider provider);
public static short ToInt16(string value, int fromBase);
public static short ToInt16(ushort value);
public static short ToInt16(uint value);
public static short ToInt16(ulong value);
public static int ToInt32(bool value);
public static int ToInt32(byte value);
public static int ToInt32(char value);
public static int ToInt32(DateTime value);
public static int ToInt32(decimal value);
public static int ToInt32(double value);
public static int ToInt32(short value);
public static int ToInt32(int value);
public static int ToInt32(long value);
public static int ToInt32(object value);
public static int ToInt32(object value, IFormatProvider provider);
public static int ToInt32(sbyte value);
public static int ToInt32(float value);
public static int ToInt32(string value);
public static int ToInt32(string value, IFormatProvider provider);
public static int ToInt32(string value, int fromBase);
public static int ToInt32(ushort value);
public static int ToInt32(uint value);
public static int ToInt32(ulong value);
public static long ToInt64(bool value);
public static long ToInt64(byte value);
public static long ToInt64(char value);

```

```
public static long ToInt64(DateTime value);
public static long ToInt64(decimal value);
public static long ToInt64(double value);
public static long ToInt64(short value);
public static long ToInt64(int value);
public static long ToInt64(long value);
public static long ToInt64(object value);
public static long ToInt64(object value, IFormatProvider provider);
public static long ToInt64(sbyte value);
public static long ToInt64(float value);
public static long ToInt64(string value);
public static long ToInt64(string value, IFormatProvider provider);
public static long ToInt64(string value, int fromBase);
public static long ToInt64(ushort value);
public static long ToInt64(uint value);
public static long ToInt64(ulong value);
public static sbyte ToSByte(bool value);
public static sbyte ToSByte(byte value);
public static sbyte ToSByte(char value);
public static sbyte ToSByte(DateTime value);
public static sbyte ToSByte(decimal value);
public static sbyte ToSByte(double value);
public static sbyte ToSByte(short value);
public static sbyte ToSByte(int value);
public static sbyte ToSByte(long value);
public static sbyte ToSByte(object value);
public static sbyte ToSByte(object value, IFormatProvider provider);
public static sbyte ToSByte(sbyte value);
public static sbyte ToSByte(float value);
public static sbyte ToSByte(string value);
public static sbyte ToSByte(string value, IFormatProvider provider);
public static sbyte ToSByte(string value, int fromBase);
public static sbyte ToSByte(ushort value);
public static sbyte ToSByte(uint value);
public static sbyte ToSByte(ulong value);
public static float ToSingle(bool value);
public static float ToSingle(byte value);
public static float ToSingle(char value);
public static float ToSingle(DateTime value);
public static float ToSingle(decimal value);
public static float ToSingle(double value);
public static float ToSingle(short value);
public static float ToSingle(int value);
public static float ToSingle(long value);
public static float ToSingle(object value);
public static float ToSingle(object value, IFormatProvider provider);
public static float ToSingle(sbyte value);
public static float ToSingle(float value);
public static float ToSingle(string value);
public static float ToSingle(string value, IFormatProvider provider);
public static float ToSingle(ushort value);
public static float ToSingle(uint value);
public static float ToSingle(ulong value);
public static string ToString(bool value);
public static string ToString(bool value, IFormatProvider provider);
```

```

public static string ToString(byte value);
public static string ToString(byte value, IFormatProvider provider);
public static string ToString(byte value, int toBase);
public static string ToString(char value);
public static string ToString(char value, IFormatProvider provider);
public static string ToString(DateTime value);
public static string ToString(DateTime value, IFormatProvider provider);
public static string ToString(decimal value);
public static string ToString(decimal value, IFormatProvider provider);
public static string ToString(double value);
public static string ToString(double value, IFormatProvider provider);
public static string ToString(short value);
public static string ToString(short value, IFormatProvider provider);
public static string ToString(short value, int toBase);
public static string ToString(int value);
public static string ToString(int value, IFormatProvider provider);
public static string ToString(int value, int toBase);
public static string ToString(long value);
public static string ToString(long value, IFormatProvider provider);
public static string ToString(long value, int toBase);
public static string ToString(object value);
public static string ToString(object value, IFormatProvider provider);
public static string ToString(sbyte value);
public static string ToString(sbyte value, IFormatProvider provider);
public static string ToString(float value);
public static string ToString(float value, IFormatProvider provider);
public static string ToString(string value);
public static string ToString(string value, IFormatProvider provider);
public static string ToString(ushort value);
public static string ToString(ushort value, IFormatProvider provider);
public static string ToString(uint value);
public static string ToString(uint value, IFormatProvider provider);
public static string ToString(ulong value);
public static string ToString(ulong value, IFormatProvider provider);
public static ushort ToUInt16(bool value);
public static ushort ToUInt16(byte value);
public static ushort ToUInt16(char value);
public static ushort ToUInt16(DateTime value);
public static ushort ToUInt16(decimal value);
public static ushort ToUInt16(double value);
public static ushort ToUInt16(short value);
public static ushort ToUInt16(int value);
public static ushort ToUInt16(long value);
public static ushort ToUInt16(object value);
public static ushort ToUInt16(object value, IFormatProvider provider);
public static ushort ToUInt16(sbyte value);
public static ushort ToUInt16(float value);
public static ushort ToUInt16(string value);
public static ushort ToUInt16(string value, IFormatProvider provider);
public static ushort ToUInt16(string value, int fromBase);
public static ushort ToUInt16(ushort value);
public static ushort ToUInt16(uint value);
public static ushort ToUInt16(ulong value);
public static uint ToUInt32(bool value);
public static uint ToUInt32(byte value);

```

```

public static uint ToUInt32(char value);
public static uint ToUInt32(DateTime value);
public static uint ToUInt32(decimal value);
public static uint ToUInt32(double value);
public static uint ToUInt32(short value);
public static uint ToUInt32(int value);
public static uint ToUInt32(long value);
public static uint ToUInt32(object value);
public static uint ToUInt32(object value, IFormatProvider provider);
public static uint ToUInt32(sbyte value);
public static uint ToUInt32(float value);
public static uint ToUInt32(string value);
public static uint ToUInt32(string value, IFormatProvider provider);
public static uint ToUInt32(string value, int fromBase);
public static uint ToUInt32(ushort value);
public static uint ToUInt32(uint value);
public static uint ToUInt32(ulong value);
public static ulong ToUInt64(bool value);
public static ulong ToUInt64(byte value);
public static ulong ToUInt64(char value);
public static ulong ToUInt64(DateTime value);
public static ulong ToUInt64(decimal value);
public static ulong ToUInt64(double value);
public static ulong ToUInt64(short value);
public static ulong ToUInt64(int value);
public static ulong ToUInt64(long value);
public static ulong ToUInt64(object value);
public static ulong ToUInt64(object value, IFormatProvider provider);
public static ulong ToUInt64(sbyte value);
public static ulong ToUInt64(float value);
public static ulong ToUInt64(string value);
public static ulong ToUInt64(string value, IFormatProvider provider);
public static ulong ToUInt64(string value, int fromBase);
public static ulong ToUInt64(ushort value);
public static ulong ToUInt64(uint value);
public static ulong ToUInt64(ulong value);
}

```

**CrossAppDomainDelegate** **delegate**  
System (mscorlib.dll) *serializable*

DESCRIPTION OF TYPE GOES HERE

```
public delegate void CrossAppDomainDelegate();
```

*Passed To:* AppDomain.DoCallback()

**DateTime** **struct**  
System (mscorlib.dll) *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public struct DateTime : IComparable, IFormattable, IConvertible {
// Public Constructors
```

```

public DateTime(int year, int month, int day);
public DateTime(int year, int month, int day, System.Globalization.Calendar calendar);
public DateTime(int year, int month, int day, int hour, int minute, int second);
public DateTime(int year, int month, int day, int hour, int minute, int second,
    System.Globalization.Calendar calendar);
public DateTime(int year, int month, int day, int hour, int minute, int second, int millisecond);
public DateTime(int year, int month, int day, int hour, int minute, int second, int millisecond,
    System.Globalization.Calendar calendar);
public DateTime(long ticks);
// Public Static Fields
public static readonly DateTime MaxValue;           =12/31/9999 11:59:59 PM
public static readonly DateTime MinValue;           =1/1/0001 12:00:00 AM
// Public Static Properties
public static DateTime Now {get; }
public static DateTime Today {get; }
public static DateTime UtcNow {get; }
// Public Instance Properties
public DateTime Date {get; }
public int Day {get; }
public DayOfWeek DayOfWeek {get; }
public int DayOfYear {get; }
public int Hour {get; }
public int Millisecond {get; }
public int Minute {get; }
public int Month {get; }
public int Second {get; }
public long Ticks {get; }
public TimeSpan TimeOfDay {get; }
public int Year {get; }
// Public Static Methods
public static int Compare(DateTime t1, DateTime t2);
public static int DaysInMonth(int year, int month);
public static bool Equals(DateTime t1, DateTime t2);
public static DateTime FromFileTime(long fileTime);
public static DateTime FromOADate(double d);
public static bool IsLeapYear(int year);
public static DateTime Parse(string s);
public static DateTime Parse(string s, IFormatProvider provider);
public static DateTime Parse(string s, IFormatProvider provider, System.Globalization.DateTimeStyles styles);
public static DateTime ParseExact(string s, string[] formats, IFormatProvider provider,
    System.Globalization.DateTimeStyles style);
public static DateTime ParseExact(string s, string format, IFormatProvider provider);
public static DateTime ParseExact(string s, string format, IFormatProvider provider,
    System.Globalization.DateTimeStyles style);
public static DateTime operator -(DateTime d, TimeSpan t);
public static TimeSpan operator -(DateTime d1, DateTime d2);
public static DateTime operator +(DateTime d, TimeSpan t);
public static bool operator !=(DateTime d1, DateTime d2);
public static bool operator <(DateTime t1, DateTime t2);
public static bool operator <=(DateTime t1, DateTime t2);
public static bool operator ==(DateTime d1, DateTime d2);
public static bool operator >(DateTime t1, DateTime t2);
public static bool operator >=(DateTime t1, DateTime t2);
// Public Instance Methods
public DateTime Add(TimeSpan value);

```

```

public DateTime AddDays(double value);
public DateTime AddHours(double value);
public DateTime AddMilliseconds(double value);
public DateTime AddMinutes(double value);
public DateTime AddMonths(int months);
public DateTime AddSeconds(double value);
public DateTime AddTicks(long value);
public DateTime AddYears(int value);
public int CompareTo(object value); // implements IComparable
public override bool Equals(object value); // overrides ValueType
public string[] GetDateTimeFormats();
public string[] GetDateTimeFormats(char format);
public string[] GetDateTimeFormats(char format, IFormatProvider provider);
public string[] GetDateTimeFormats(IFormatProvider provider);
public override int GetHashCode(); // overrides ValueType
public TypeCode GetTypeCode(); // implements IConvertible
public DateTime Subtract(TimeSpan value);
public TimeSpan Subtract(DateTime value);
public long ToFileTime();
public DateTime ToLocalTime();
public string ToLongDateString();
public string ToLongTimeString();
public double ToOADate();
public string ToShortDateString();
public string ToShortTimeString();
public override string ToString(); // overrides ValueType
public string ToString(IFormatProvider provider); // implements IConvertible
public string ToString(string format);
public string ToString(string format, IFormatProvider provider); // implements IFormattable
public DateTime ToUniversalTime();
}

```

*Hierarchy:* Object → ValueType → DateTime(IComparable, IFormattable, IConvertible)

*Returned By:* Multiple types

*Passed To:* Multiple types

## DayOfWeek

enum

System (mscorlib.dll)

serializable

DESCRIPTION OF TYPE GOES HERE

```

public enum DayOfWeek {
    Sunday = 0,
    Monday = 1,
    Tuesday = 2,
    Wednesday = 3,
    Thursday = 4,
    Friday = 5,
    Saturday = 6
}

```

*Hierarchy:* Object → ValueType → Enum(IComparable, IFormattable, IConvertible) → DayOfWeek

*Returned By:* DateTime.DayOfWeek, System.Globalization.Calendar.GetDayOfWeek(), System.Globalization.DateTimeFormatInfo.FirstDayOfWeek

*Passed To:* System.Globalization.Calendar.GetWeekOfYear(),  
System.Globalization.DateTimeFormatInfo.{FirstDayOfWeek, GetAbbreviatedDayName(), GetDayName()}

## **DBNull** **sealed class**

**System (mscorlib.dll)** **serializable**

DESCRIPTION OF TYPE GOES HERE

```
public sealed class DBNull : System.Runtime.Serialization.IEnumerable, IConvertible {  
    // Public Static Fields  
    public static readonly DBNull Value;  
    // Public Instance Methods  
    public void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // implements ISerializable  
        System.Runtime.Serialization.StreamingContext context);  
    public TypeCode GetTypeCode(); // implements IConvertible  
    public override string ToString(); // overrides object  
    public string ToString(IFormatProvider provider); // implements IConvertible  
}
```

## **Decimal** **struct**

**System (mscorlib.dll)** **ECMA, serializable**

DESCRIPTION OF TYPE GOES HERE

```
public struct Decimal : IFormattable, IComparable, IConvertible {  
    // Public Constructors  
    public Decimal(double value);  
    public Decimal(int value);  
    public Decimal(int[] bits);  
    public Decimal(int lo, int mid, int hi, bool isNegative, byte scale);  
    public Decimal(long value);  
    public Decimal(float value);  
    public Decimal(uint value);  
    public Decimal(ulong value);  
    // Public Static Fields  
    public static readonly decimal MaxValue; // =79228162514264337593543950335  
    public static readonly decimal MinusOne; // =-1  
    public static readonly decimal MinValue; // =-79228162514264337593543950335  
    public static readonly decimal One; // =1  
    public static readonly decimal Zero; // =0  
    // Public Static Methods  
    public static decimal Add(decimal d1, decimal d2);  
    public static int Compare(decimal d1, decimal d2);  
    public static decimal Divide(decimal d1, decimal d2);  
    public static bool Equals(decimal d1, decimal d2);  
    public static decimal Floor(decimal d);  
    public static decimal FromOACurrency(long cy);  
    public static int[] GetBits(decimal d);  
    public static decimal Multiply(decimal d1, decimal d2);  
    public static decimal Negate(decimal d);  
    public static decimal Parse(string s);  
    public static decimal Parse(string s, IFormatProvider provider);  
    public static decimal Parse(string s, System.Globalization.NumberStyles style);  
    public static decimal Parse(string s, System.Globalization.NumberStyles style, IFormatProvider provider);  
    public static decimal Remainder(decimal d1, decimal d2);  
    public static decimal Round(decimal d, int decimals);  
    public static decimal Subtract(decimal d1, decimal d2);
```

```

public static byte ToByte(decimal value);
public static double ToDouble(decimal d);
public static short ToInt16(decimal value);
public static int ToInt32(decimal d);
public static long ToInt64(decimal d);
public static long ToOACurrency(decimal value);
public static sbyte ToSByte(decimal value);
public static float ToSingle(decimal d);
public static ushort ToUInt16(decimal value);
public static uint ToUInt32(decimal d);
public static ulong ToUInt64(decimal d);
public static decimal Truncate(decimal d);
public static decimal operator %(decimal d1, decimal d2);
public static decimal operator *(decimal d1, decimal d2);
public static decimal operator /(decimal d1, decimal d2);
public static decimal operator --(decimal d);
public static decimal operator -(decimal d);
public static decimal operator -(decimal d1, decimal d2);
public static decimal operator +(decimal d);
public static decimal operator +(decimal d1, decimal d2);
public static decimal operator ++(decimal d);
public static bool operator !=(decimal d1, decimal d2);
public static bool operator <(decimal d1, decimal d2);
public static bool operator <=(decimal d1, decimal d2);
public static bool operator ==(decimal d1, decimal d2);
public static bool operator >(decimal d1, decimal d2);
public static bool operator >=(decimal d1, decimal d2);
public static explicit operator byte(decimal value);
public static explicit operator char(decimal value);
public static explicit operator decimal(double value);
public static explicit operator decimal(float value);
public static explicit operator double(decimal value);
public static explicit operator short(decimal value);
public static explicit operator int(decimal value);
public static explicit operator long(decimal value);
public static explicit operator sbyte(decimal value);
public static explicit operator float(decimal value);
public static explicit operator ushort(decimal value);
public static explicit operator uint(decimal value);
public static explicit operator ulong(decimal value);
public static implicit operator decimal(byte value);
public static implicit operator decimal(char value);
public static implicit operator decimal(short value);
public static implicit operator decimal(int value);
public static implicit operator decimal(long value);
public static implicit operator decimal(sbyte value);
public static implicit operator decimal(ushort value);
public static implicit operator decimal(uint value);
public static implicit operator decimal(ulong value);
// Public Instance Methods
public int CompareTo(object value); // implements IComparable
public override bool Equals(object value); // overrides ValueType
public override int GetHashCode(); // overrides ValueType
public TypeCode GetTypeCode(); // implements IConvertible
public override string ToString(); // overrides ValueType

```

```

public string ToString(IFormatProvider provider); // implements IConvertible
public string ToString(string format);
public string ToString(string format, IFormatProvider provider); // implements IFormattable
}

```

*Hierarchy:* Object → ValueType → Decimal(IFormattable, IComparable, IConvertible)

*Returned By:* Convert.ToDecimal(), IConvertible.ToDecimal(), System.IO.BinaryReader.ReadDecimal(), System.Runtime.Serialization.FormatterConverter.ToDecimal(), System.Runtime.Serialization.FormatterConverter.ToDecimal(), System.Runtime.Serialization.SerializationInfo.GetDecimal(), System.Xml.XmlConvert.ToDecimal()

*Passed To:* Multiple types

## Delegate

**abstract class**

**System (mscorlib.dll)**

**ECMA, serializable**

DESCRIPTION OF TYPE GOES HERE

```

public abstract class Delegate : ICloneable, System.Runtime.Serialization.ISerializable {
// Protected Constructors
protected Delegate(object target, string method);
protected Delegate(Type target, string method);
// Public Instance Properties
public MethodInfo Method {get; }
public object Target {get; }
// Public Static Methods
public static Delegate Combine(Delegate[] delegates);
public static Delegate Combine(Delegate a, Delegate b);
public static Delegate CreateDelegate(Type type, System.Reflection.MethodInfo method);
public static Delegate CreateDelegate(Type type, object target, string method);
public static Delegate CreateDelegate(Type type, object target, string method, bool ignoreCase);
public static Delegate CreateDelegate(Type type, Type target, string method);
public static Delegate Remove(Delegate source, Delegate value);
public static bool operator !=(Delegate d1, Delegate d2);
public static bool operator ==(Delegate d1, Delegate d2);
// Public Instance Methods
public virtual object Clone(); // implements ICloneable
public object DynamicInvoke(object[] args);
public override bool Equals(object obj); // overrides object
public override int GetHashCode(); // overrides object
public virtual Delegate[] GetInvocationList();
public virtual void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // implements ISerializable
System.Runtime.Serialization.StreamingContext context);
// Protected Instance Methods
protected virtual Delegate CombineImpl(Delegate d);
protected virtual object DynamicInvokeImpl(object[] args);
protected virtual MethodInfo GetMethodImpl();
protected virtual Delegate RemoveImpl(Delegate d);
}

```

*Subclasses:* MulticastDelegate

*Returned By:* MulticastDelegate.{CombineImpl(), GetInvocationList(), RemoveImpl()}

*Passed To:* MulticastDelegate.{CombineImpl(), RemoveImpl()}, System.Reflection.EventInfo.{AddEventHandler(), RemoveEventHandler()}, System.Runtime.InteropServices.Expando.IExpando.AddMethod()

**DivideByZeroException****class**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class DivideByZeroException : ArithmeticException {
// Public Constructors
public DivideByZeroException();
public DivideByZeroException(string message);
public DivideByZeroException(string message, Exception innerException);
// Protected Constructors
protected DivideByZeroException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → ArithmeticException → DivideByZeroException

**DIINotFoundException****class**

System (mscorlib.dll)

serializable

DESCRIPTION OF TYPE GOES HERE

```
public class DIINotFoundException : TypeLoadException {
// Public Constructors
public DIINotFoundException();
public DIINotFoundException(string message);
public DIINotFoundException(string message, Exception inner);
// Protected Constructors
protected DIINotFoundException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → TypeLoadException → DIINotFoundException

**Double****struct**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct Double : IComparable, IFormattable, IConvertible {
// Public Static Fields
public const double Epsilon; // =4.94065645841247E-324
public const double MaxValue; // =1.79769313486232E+308
public const double MinValue; // =-1.79769313486232E+308
public const double NaN; // =NaN
public const double NegativeInfinity; // =-Infinity
public const double PositiveInfinity; // =Infinity
// Public Static Methods
public static bool IsInfinity(double d);
public static bool IsNaN(double d);
public static bool IsNegativeInfinity(double d);
public static bool IsPositiveInfinity(double d);
public static double Parse(string s);
public static double Parse(string s, IFormatProvider provider);
public static double Parse(string s, System.Globalization.NumberStyles style);
public static double Parse(string s, System.Globalization.NumberStyles style, IFormatProvider provider);
}
```

```

public static bool TryParse(string s, System.Globalization.NumberStyles style, IFormatProvider provider,
                             out double result);
// Public Instance Methods
public int CompareTo(object value); // implements IComparable
public override bool Equals(object obj); // overrides ValueType
public override int GetHashCode(); // overrides ValueType
public TypeCode GetTypeCode(); // implements IConvertible
public override string ToString(); // overrides ValueType
public string ToString(IFormatProvider provider); // implements IConvertible
public string ToString(string format);
public string ToString(string format, IFormatProvider provider); // implements IFormattable
}

```

*Hierarchy:* Object → ValueType → Double(IComparable, IFormattable, IConvertible)

*Returned By:* Multiple types

*Passed To:* Multiple types

## DuplicateWaitObjectException

**class**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class DuplicateWaitObjectException : ArgumentException {
// Public Constructors
public DuplicateWaitObjectException();
public DuplicateWaitObjectException(string parameterName);
public DuplicateWaitObjectException(string parameterName, string message);
// Protected Constructors
protected DuplicateWaitObjectException(System.Runtime.Serialization.SerializationInfo info,
                                         System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → ArgumentException → DuplicateWaitObjectException

## EntryPointNotFoundException

**class**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class EntryPointNotFoundException : TypeLoadException {
// Public Constructors
public EntryPointNotFoundException();
public EntryPointNotFoundException(string message);
public EntryPointNotFoundException(string message, Exception inner);
// Protected Constructors
protected EntryPointNotFoundException(System.Runtime.Serialization.SerializationInfo info,
                                         System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → TypeLoadException → EntryPointNotFoundException

## Enum

**abstract class**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public abstract class Enum : ValueType, IComparable, IFormattable, IConvertible {
// Protected Constructors
protected Enum();
// Public Static Methods
public static string Format(Type enumType, object value, string format);
public static string GetName(Type enumType, object value);
public static string[] GetNames(Type enumType);
public static Type GetUnderlyingType(Type enumType);
public static Array GetValues(Type enumType);
public static bool IsDefined(Type enumType, object value);
public static object Parse(Type enumType, string value);
public static object Parse(Type enumType, string value, bool ignoreCase);
public static object ToObject(Type enumType, byte value);
public static object ToObject(Type enumType, short value);
public static object ToObject(Type enumType, int value);
public static object ToObject(Type enumType, long value);
public static object ToObject(Type enumType, object value);
public static object ToObject(Type enumType, sbyte value);
public static object ToObject(Type enumType, ushort value);
public static object ToObject(Type enumType, uint value);
public static object ToObject(Type enumType, ulong value);
// Public Instance Methods
public int CompareTo(object target); // implements IComparable
public override bool Equals(object obj); // overrides ValueType
public override int GetHashCode(); // overrides ValueType
public TypeCode GetTypeCode(); // implements IConvertible
public override string ToString(); // overrides ValueType
public string ToString(IFormatProvider provider); // implements IConvertible
public string ToString(string format);
public string ToString(string format, IFormatProvider provider); // implements IFormattable
}
```

*Hierarchy:* Object → ValueType → Enum(IComparable, IFormattable, IConvertible)

*Subclasses:* Multiple types

## Environment

sealed class

System (mscorlib.dll)

ECMA

DESCRIPTION OF TYPE GOES HERE

```
public sealed class Environment {
// Public Static Properties
public static string CommandLine {get; }
public static string CurrentDirectory {set; get; }
public static int ExitCode {set; get; }
public static string NewLine {get; }
public static string StackTrace {get; }
public static string SystemDirectory {get; }
public static int TickCount {get; }
public static Version Version {get; }
public static long WorkingSet {get; }
// Public Instance Properties
public bool HasShutdownStarted {get; }
// Public Static Methods
public static void Exit(int exitCode);
}
```

```

public static string[] GetCommandLineArgs();
public static string GetEnvironmentVariable(string variable);
public static IDictionary GetEnvironmentVariables();
public static string[] GetLogicalDrives();
}

```

---

## **EventArgs** class

**System (mscorlib.dll)** *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```

public class EventArgs {
// Public Constructors
public EventArgs();
// Public Static Fields
public static readonly EventArgs Empty; =System.EventArgs
}

```

*Subclasses:* AssemblyLoadEventArgs, ResolveEventArgs, UnhandledExceptionEventArgs

*Passed To:* EventHandler.BeginInvoke(), Invoke()

---

## **EventHandler** delegate

**System (mscorlib.dll)** *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```

public delegate void EventHandler(object sender, EventArgs e);

```

*Associated Events:* AppDomain.DomainUnload(), ProcessExit(),  
System.Diagnostics.Process.Disposed(), Exited(), System.Net.WebClient.Disposed()

---

## **Exception** class

**System (mscorlib.dll)** *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```

public class Exception : System.Runtime.Serialization.ISerializable {
// Public Constructors
public Exception();
public Exception(string message);
public Exception(string message, Exception innerException);
// Protected Constructors
protected Exception(System.Runtime.Serialization.SerializationInfo info,
                    System.Runtime.Serialization.StreamingContext context);
// Public Instance Properties
public virtual string HelpLink {set; get; }
public Exception InnerException {get; }
public virtual string Message {get; }
public virtual string Source {set; get; }
public virtual string StackTrace {get; }
public MethodBase TargetSite {get; }
// Protected Instance Properties
protected int HResult {set; get; }
// Public Instance Methods
public virtual Exception GetBaseException();
}

```

```
public virtual void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // implements ISerializable
    System.Runtime.Serialization.StreamingContext context);
public override string ToString(); // overrides object
}
```

*Subclasses:* ApplicationException, SystemException,  
System.IO.IsolatedStorage.IsolatedStorageException

*Returned By:* System.Reflection.ReflectionTypeLoadException.LoaderExceptions

*Passed To:* Multiple types

## **ExecutionEngineException** sealed class

**System (mscorlib.dll)** ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public sealed class ExecutionEngineException : SystemException {
// Public Constructors
public ExecutionEngineException();
public ExecutionEngineException(string message);
public ExecutionEngineException(string message, Exception innerException);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException →  
ExecutionEngineException

## **FieldAccessException** class

**System (mscorlib.dll)** ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class FieldAccessException : MemberAccessException {
// Public Constructors
public FieldAccessException();
public FieldAccessException(string message);
public FieldAccessException(string message, Exception inner);
// Protected Constructors
protected FieldAccessException(System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException →  
MemberAccessException → FieldAccessException

## **FlagsAttribute** class

**System (mscorlib.dll)** ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class FlagsAttribute : Attribute {
// Public Constructors
public FlagsAttribute();
}
```

*Hierarchy:* Object → Attribute → FlagsAttribute

*Valid On:* Enum

## **FormatException** class

---

**System (mscorlib.dll)**

*ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public class FormatException : SystemException {  
    // Public Constructors  
    public FormatException();  
    public FormatException(string message);  
    public FormatException(string message, Exception innerException);  
    // Protected Constructors  
    protected FormatException(System.Runtime.Serialization.SerializationInfo info,  
        System.Runtime.Serialization.StreamingContext context);  
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → FormatException

*Subclasses:* UriFormatException, System.Net.CookieException, System.Reflection.CustomAttributeFormatException

---

**GC**

**sealed class**

**System (mscorlib.dll)**

*ECMA*

DESCRIPTION OF TYPE GOES HERE

```
public sealed class GC {  
    // Public Static Properties  
    public static int MaxGeneration {get; }  
    // Public Static Methods  
    public static void Collect();  
    public static void Collect(int generation);  
    public static int GetGeneration(object obj);  
    public static int GetGeneration(WeakReference wo);  
    public static long GetTotalMemory(bool forceFullCollection);  
    public static void KeepAlive(object obj);  
    public static void ReRegisterForFinalize(object obj);  
    public static void SuppressFinalize(object obj);  
    public static void WaitForPendingFinalizers();  
}
```

---

**Guid**

**struct**

**System (mscorlib.dll)**

*serializable*

DESCRIPTION OF TYPE GOES HERE

```
public struct Guid : IFormattable, IComparable {  
    // Public Constructors  
    public Guid(byte[] b);  
    public Guid(int a, short b, short c, byte[] d);  
    public Guid(int a, short b, short c, byte d, byte e, byte f, byte g, byte h, byte i, byte j, byte k);  
    public Guid(string g);  
    public Guid(uint a, ushort b, ushort c, byte d, byte e, byte f, byte g, byte h, byte i, byte j, byte k);  
    // Public Static Fields  
    public static readonly Guid Empty; =00000000-0000-0000-0000-000000000000  
    // Public Static Methods  
    public static Guid NewGuid();  
    public static bool operator !=(Guid a, Guid b);  
    public static bool operator ==(Guid a, Guid b);  
}
```

```
// Public Instance Methods
public int CompareTo(object value); // implements IComparable
public override bool Equals(object o); // overrides ValueType
public override int GetHashCode(); // overrides ValueType
public byte[] ToByteArray();
public override string ToString(); // overrides ValueType
public string ToString(string format);
public string ToString(string format, IFormatProvider provider); // implements IFormattable
}
```

*Hierarchy:* Object → ValueType → Guid(IFormattable, IComparable)

*Returned By:* System.Reflection.Emit.UnmanagedMarshal.IIDGuid, Type.GUID,  
System.Xml.XmlConvert.ToGuid()

*Passed To:* System.Reflection.Emit.ModuleBuilder.DefineDocument(),  
System.Xml.XmlConvert.ToString()

## **IAsyncResult** interface

**System (mscorlib.dll)** ECMA

DESCRIPTION OF TYPE GOES HERE

```
public interface IAsyncResult {
// Public Instance Properties
public object AsyncState {get; }
public WaitHandle AsyncWaitHandle {get; }
public bool CompletedSynchronously {get; }
public bool IsCompleted {get; }
}
```

*Returned By:* Multiple types

*Passed To:* Multiple types

## **ICloneable** interface

**System (mscorlib.dll)** ECMA

DESCRIPTION OF TYPE GOES HERE

```
public interface ICloneable {
// Public Instance Methods
public object Clone();
}
```

*Implemented By:* Multiple types

## **IComparable** interface

**System (mscorlib.dll)** ECMA

DESCRIPTION OF TYPE GOES HERE

```
public interface IComparable {
// Public Instance Methods
public int CompareTo(object obj);
}
```

*Implemented By:* Multiple types

## **IConvertible** interface

---

**System (mscorlib.dll)**

DESCRIPTION OF TYPE GOES HERE

```
public interface IConvertible {  
    // Public Instance Methods  
    public TypeCode GetTypeCode();  
    public bool ToBoolean(IFormatProvider provider);  
    public byte ToByte(IFormatProvider provider);  
    public char ToChar(IFormatProvider provider);  
    public DateTime ToDateTime(IFormatProvider provider);  
    public decimal ToDecimal(IFormatProvider provider);  
    public double ToDouble(IFormatProvider provider);  
    public short ToInt16(IFormatProvider provider);  
    public int ToInt32(IFormatProvider provider);  
    public long ToInt64(IFormatProvider provider);  
    public sbyte ToSByte(IFormatProvider provider);  
    public float ToSingle(IFormatProvider provider);  
    public string Tostring(IFormatProvider provider);  
    public object ToType(Type conversionType, IFormatProvider provider);  
    public ushort ToUInt16(IFormatProvider provider);  
    public uint ToUInt32(IFormatProvider provider);  
    public ulong ToUInt64(IFormatProvider provider);  
}
```

*Implemented By:* Multiple types

---

**ICustomFormatter****interface****System (mscorlib.dll)**

DESCRIPTION OF TYPE GOES HERE

```
public interface ICustomFormatter {  
    // Public Instance Methods  
    public string Format(string format, object arg, IFormatProvider formatProvider);  
}
```

---

**IDisposable****interface****System (mscorlib.dll)***ECMA*

DESCRIPTION OF TYPE GOES HERE

```
public interface IDisposable {  
    // Public Instance Methods  
    public void Dispose();  
}
```

*Implemented By:* Multiple types

---

**IFormatProvider****interface****System (mscorlib.dll)***ECMA*

DESCRIPTION OF TYPE GOES HERE

```
public interface IFormatProvider {  
    // Public Instance Methods  
    public object GetFormat(Type formatType);  
}
```

*Implemented By:* System.Globalization.{CultureInfo, DateTimeFormatInfo, NumberFormatInfo}

*Returned By:* System.IO.TextWriter.FormatProvider

*Passed To:* Multiple types

**IFormattable** interface  
 System (mscorlib.dll) ECMA

DESCRIPTION OF TYPE GOES HERE

```
public interface IFormattable {
// Public Instance Methods
  public string ToString(string format, IFormatProvider formatProvider);
}
```

*Implemented By:* Multiple types

**IndexOutOfRangeException** sealed class  
 System (mscorlib.dll) ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public sealed class IndexOutOfRangeException : SystemException {
// Public Constructors
  public IndexOutOfRangeException();
  public IndexOutOfRangeException(string message);
  public IndexOutOfRangeException(string message, Exception innerException);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → IndexOutOfRangeException

**Int16** struct  
 System (mscorlib.dll) ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct Int16 : IComparable, IFormattable, IConvertible {
// Public Static Fields
  public const short MaxValue; // =32767
  public const short MinValue; // =-32768
// Public Static Methods
  public static short Parse(string s);
  public static short Parse(string s, IFormatProvider provider);
  public static short Parse(string s, System.Globalization.NumberStyles style);
  public static short Parse(string s, System.Globalization.NumberStyles style, IFormatProvider provider);
// Public Instance Methods
  public int CompareTo(object value); // implements IComparable
  public override bool Equals(object obj); // overrides ValueType
  public override int GetHashCode(); // overrides ValueType
  public TypeCode GetTypeCode(); // implements IConvertible
  public override string ToString(); // overrides ValueType
  public string ToString(IFormatProvider provider); // implements IConvertible
  public string ToString(string format);
  public string ToString(string format, IFormatProvider provider); // implements IFormattable
}
```

*Hierarchy:* Object → ValueType → Int16(IComparable, IFormattable, IConvertible)

*Returned By:* Multiple types

*Passed To:* Multiple types

## Int32

**struct**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct Int32 : IComparable, IFormattable, IConvertible {  
    // Public Static Fields  
    public const int MaxValue; // =2147483647  
    public const int MinValue; // =-2147483648  
    // Public Static Methods  
    public static int Parse(string s);  
    public static int Parse(string s, IFormatProvider provider);  
    public static int Parse(string s, System.Globalization.NumberStyles style);  
    public static int Parse(string s, System.Globalization.NumberStyles style, IFormatProvider provider);  
    // Public Instance Methods  
    public int CompareTo(object value); // implements IComparable  
    public override bool Equals(object obj); // overrides ValueType  
    public override int GetHashCode(); // overrides ValueType  
    public TypeCode GetTypeCode(); // implements IConvertible  
    public override string ToString(); // overrides ValueType  
    public string ToString(IFormatProvider provider); // implements IConvertible  
    public string ToString(string format);  
    public string ToString(string format, IFormatProvider provider); // implements IFormattable  
}
```

*Hierarchy:* Object → ValueType → Int32(IComparable, IFormattable, IConvertible)

*Returned By:* Multiple types

*Passed To:* Multiple types

## Int64

**struct**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct Int64 : IComparable, IFormattable, IConvertible {  
    // Public Static Fields  
    public const long MaxValue; // =9223372036854775807  
    public const long MinValue; // =-9223372036854775808  
    // Public Static Methods  
    public static long Parse(string s);  
    public static long Parse(string s, IFormatProvider provider);  
    public static long Parse(string s, System.Globalization.NumberStyles style);  
    public static long Parse(string s, System.Globalization.NumberStyles style, IFormatProvider provider);  
    // Public Instance Methods  
    public int CompareTo(object value); // implements IComparable  
    public override bool Equals(object obj); // overrides ValueType  
    public override int GetHashCode(); // overrides ValueType  
    public TypeCode GetTypeCode(); // implements IConvertible  
    public override string ToString(); // overrides ValueType  
    public string ToString(IFormatProvider provider); // implements IConvertible  
    public string ToString(string format);  
    public string ToString(string format, IFormatProvider provider); // implements IFormattable  
}
```

*Hierarchy:* Object → ValueType → Int64(IComparable, IFormattable, IConvertible)

*Returned By:* Multiple types

*Passed To:* Multiple types

## IntPtr

**struct**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct IntPtr : System.Runtime.Serialization.ISerializable {
// Public Constructors
    public IntPtr(int value);
    public IntPtr(long value);
    public IntPtr(void *value);
// Public Static Fields
    public static readonly IntPtr Zero; // -0
// Public Static Properties
    public static int Size {get; }
// Public Static Methods
    public static bool operator !=(IntPtr value1, IntPtr value2);
    public static bool operator ==(IntPtr value1, IntPtr value2);
    public static explicit operator int(IntPtr value);
    public static explicit operator long(IntPtr value);
    public static explicit operator IntPtr(int value);
    public static explicit operator IntPtr(long value);
    public static explicit operator IntPtr(void *value);
    public static explicit operator Void(IntPtr value);
// Public Instance Methods
    public override bool Equals(object obj); // overrides ValueType
    public override int GetHashCode(); // overrides ValueType
    public int.ToInt32();
    public long.ToInt64();
    public void* ToPointer();
    public override string ToString(); // overrides ValueType
    public string ToString(string format);
}
```

*Hierarchy:* Object → ValueType → IntPtr(System.Runtime.Serialization.ISerializable)

*Returned By:* Multiple types

*Passed To:* Multiple types

## InvalidCastException

**class**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class InvalidCastException : SystemException {
// Public Constructors
    public InvalidCastException();
    public InvalidCastException(string message);
    public InvalidCastException(string message, Exception innerException);
// Protected Constructors
    protected InvalidCastException(System.Runtime.Serialization.SerializationInfo info,
        System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → InvalidCastException

## **InvalidOperationException** **class**

System (mscorlib.dll) *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public class InvalidOperationException : SystemException {  
    // Public Constructors  
    public InvalidOperationException();  
    public InvalidOperationException(string message);  
    public InvalidOperationException(string message, Exception innerException);  
    // Protected Constructors  
    protected InvalidOperationException(System.Runtime.Serialization.SerializationInfo info,  
        System.Runtime.Serialization.StreamingContext context);  
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → InvalidOperationException

*Subclasses:* ObjectDisposedException, System.Net.{ProtocolViolationException, WebException}

## **InvalidProgramException** **sealed class**

System (mscorlib.dll) *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public sealed class InvalidProgramException : SystemException {  
    // Public Constructors  
    public InvalidProgramException();  
    public InvalidProgramException(string message);  
    public InvalidProgramException(string message, Exception inner);  
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → InvalidProgramException

## **IServiceProvider** **interface**

System (mscorlib.dll)

DESCRIPTION OF TYPE GOES HERE

```
public interface IServiceProvider {  
    // Public Instance Methods  
    public object GetService(Type serviceType);  
}
```

## **LoaderOptimization** **enum**

System (mscorlib.dll) *serializable*

DESCRIPTION OF TYPE GOES HERE

```
public enum LoaderOptimization {  
    NotSpecified = 0,  
    SingleDomain = 1,  
    MultiDomain = 2,  
    MultiDomainHost = 3  
}
```

}

*Hierarchy:* Object → ValueType → Enum(Comparable, Formattable, Convertible) → LoaderOptimization

*Returned By:* AppDomainSetup.LoaderOptimization, LoaderOptimizationAttribute.Value

*Passed To:* AppDomainSetup.LoaderOptimization, LoaderOptimizationAttribute.LoaderOptimizationAttribute()

## LoaderOptimizationAttribute

sealed class

System (mscorlib.dll)

DESCRIPTION OF TYPE GOES HERE

```
public sealed class LoaderOptimizationAttribute : Attribute {
// Public Constructors
    public LoaderOptimizationAttribute(byte value);
    public LoaderOptimizationAttribute(LoaderOptimization value);
// Public Instance Properties
    public LoaderOptimization Value {get;}
}
```

*Hierarchy:* Object → Attribute → LoaderOptimizationAttribute

*Valid On:* Method

## LocalDataStoreSlot

sealed class

System (mscorlib.dll)

DESCRIPTION OF TYPE GOES HERE

```
public sealed class LocalDataStoreSlot {
// Protected Instance Methods
    protected override void Finalize(); // overrides object
}
```

*Returned By:* System.Threading.Thread.{AllocateDataSlot(), AllocateNamedDataSlot(), GetNamedDataSlot()}

*Passed To:* System.Threading.Thread.{GetData(), SetData()}

## MarshalByRefObject

abstract class

System (mscorlib.dll)

ECMA, serializable, marshal by reference

DESCRIPTION OF TYPE GOES HERE

```
public abstract class MarshalByRefObject {
// Protected Constructors
    protected MarshalByRefObject();
// Public Instance Methods
    public virtual ObjRef CreateObjRef(Type requestedType);
    public object GetLifetimeService();
    public virtual object InitializeLifetimeService();
}
```

*Subclasses:* Multiple types

## Math

sealed class

System (mscorlib.dll)

ECMA

DESCRIPTION OF TYPE GOES HERE

```
public sealed class Math {  
    // Public Static Fields  
    public const double E;                =2.71828182845905  
    public const double PI;              =3.14159265358979  
    // Public Static Methods  
    public static decimal Abs(decimal value);  
    public static double Abs(double value);  
    public static short Abs(short value);  
    public static int Abs(int value);  
    public static long Abs(long value);  
    public static sbyte Abs(sbyte value);  
    public static float Abs(float value);  
    public static double Acos(double d);  
    public static double Asin(double d);  
    public static double Atan(double d);  
    public static double Atan2(double y, double x);  
    public static double Ceiling(double a);  
    public static double Cos(double d);  
    public static double Cosh(double value);  
    public static double Exp(double d);  
    public static double Floor(double d);  
    public static double IEEERemainder(double x, double y);  
    public static double Log(double d);  
    public static double Log(double a, double newBase);  
    public static double Log10(double d);  
    public static byte Max(byte val1, byte val2);  
    public static decimal Max(decimal val1, decimal val2);  
    public static double Max(double val1, double val2);  
    public static short Max(short val1, short val2);  
    public static int Max(int val1, int val2);  
    public static long Max(long val1, long val2);  
    public static sbyte Max(sbyte val1, sbyte val2);  
    public static float Max(float val1, float val2);  
    public static ushort Max(ushort val1, ushort val2);  
    public static uint Max(uint val1, uint val2);  
    public static ulong Max(ulong val1, ulong val2);  
    public static byte Min(byte val1, byte val2);  
    public static decimal Min(decimal val1, decimal val2);  
    public static double Min(double val1, double val2);  
    public static short Min(short val1, short val2);  
    public static int Min(int val1, int val2);  
    public static long Min(long val1, long val2);  
    public static sbyte Min(sbyte val1, sbyte val2);  
    public static float Min(float val1, float val2);  
    public static ushort Min(ushort val1, ushort val2);  
    public static uint Min(uint val1, uint val2);  
    public static ulong Min(ulong val1, ulong val2);  
    public static double Pow(double x, double y);  
    public static decimal Round(decimal d);  
    public static decimal Round(decimal d, int decimals);  
    public static double Round(double a);  
    public static double Round(double value, int digits);  
    public static int Sign(decimal value);  
    public static int Sign(double value);  
    public static int Sign(short value);  
}
```

```

public static int Sign(int value);
public static int Sign(long value);
public static int Sign(sbyte value);
public static int Sign(float value);
public static double Sin(double a);
public static double Sinh(double value);
public static double Sqrt(double d);
public static double Tan(double a);
public static double Tanh(double value);
}

```

### MemberAccessException

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class MemberAccessException : SystemException {
// Public Constructors
public MemberAccessException();
public MemberAccessException(string message);
public MemberAccessException(string message, Exception inner);
// Protected Constructors
protected MemberAccessException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → MemberAccessException

*Subclasses:* FieldAccessException, MethodAccessException, MissingMemberException

### MethodAccessException

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class MethodAccessException : MemberAccessException {
// Public Constructors
public MethodAccessException();
public MethodAccessException(string message);
public MethodAccessException(string message, Exception inner);
// Protected Constructors
protected MethodAccessException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → MemberAccessException → MethodAccessException

### MissingFieldException

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class MissingFieldException : MissingMemberException {
// Public Constructors
public MissingFieldException();
public MissingFieldException(string message);
}

```

```

public MissingFieldException(string message, Exception inner);
public MissingFieldException(string className, string fieldName);
// Protected Constructors
protected MissingFieldException(System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
// Public Instance Properties
public override string Message {get; } // overrides MissingMemberException
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → MemberAccessException → MissingMemberException → MissingFieldException

## MissingMemberException

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class MissingMemberException : MemberAccessException {
// Public Constructors
public MissingMemberException();
public MissingMemberException(string message);
public MissingMemberException(string message, Exception inner);
public MissingMemberException(string className, string memberName);
// Protected Constructors
protected MissingMemberException(System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
// Protected Instance Fields
protected string ClassName;
protected string MemberName;
protected byte[] Signature;
// Public Instance Properties
public override string Message {get; } // overrides Exception
// Public Instance Methods
public override void GetObjectData(System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → MemberAccessException → MissingMemberException

*Subclasses:* MissingFieldException, MissingMethodException

## MissingMethodException

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class MissingMethodException : MissingMemberException {
// Public Constructors
public MissingMethodException();
public MissingMethodException(string message);
public MissingMethodException(string message, Exception inner);
public MissingMethodException(string className, string methodName);
// Protected Constructors
protected MissingMethodException(System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
// Public Instance Properties

```

```
public override string Message {get;} // overrides MissingMemberException
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → MemberAccessException → MissingMemberException → MissingMethodException

**MulticastDelegate** **abstract class**  
**System (mscorlib.dll)** **serializable**

DESCRIPTION OF TYPE GOES HERE

```
public abstract class MulticastDelegate : Delegate {
// Protected Constructors
protected MulticastDelegate(object target, string method);
protected MulticastDelegate(Type target, string method);
// Public Static Methods
public static bool operator !=(MulticastDelegate d1, MulticastDelegate d2);
public static bool operator ==(MulticastDelegate d1, MulticastDelegate d2);
// Public Instance Methods
public sealed override bool Equals(object obj); // overrides Delegate
public sealed override int GetHashCode(); // overrides Delegate
public sealed override Delegate[] GetInvocationList(); // overrides Delegate
public override void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // overrides Delegate
System.Runtime.Serialization.StreamingContext context);
// Protected Instance Methods
protected sealed override Delegate CombineImpl(Delegate follow); // overrides Delegate
protected sealed override object DynamicInvokeImpl(object[] args); // overrides Delegate
protected sealed override Delegate RemoveImpl(Delegate value); // overrides Delegate
}
```

*Hierarchy:* Object → Delegate(ICloneable, System.Runtime.Serialization.ISerializable) → MulticastDelegate

*Subclasses:* Multiple types

**MulticastNotSupportedException** **sealed class**  
**System (mscorlib.dll)** **serializable**

DESCRIPTION OF TYPE GOES HERE

```
public sealed class MulticastNotSupportedException : SystemException {
// Public Constructors
public MulticastNotSupportedException();
public MulticastNotSupportedException(string message);
public MulticastNotSupportedException(string message, Exception inner);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → MulticastNotSupportedException

**NonSerializedAttribute** **sealed class**  
**System (mscorlib.dll)**

DESCRIPTION OF TYPE GOES HERE

```
public sealed class NonSerializedAttribute : Attribute {
// Public Constructors
public NonSerializedAttribute();
}
```

}

*Hierarchy:* Object → Attribute → NonSerializedAttribute

*Valid On:* Field

## NotFiniteNumberException

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class NotFiniteNumberException : ArithmeticException {  
    // Public Constructors  
    public NotFiniteNumberException();  
    public NotFiniteNumberException(double offendingNumber);  
    public NotFiniteNumberException(string message);  
    public NotFiniteNumberException(string message, double offendingNumber);  
    public NotFiniteNumberException(string message, double offendingNumber, Exception innerException);  
    // Protected Constructors  
    protected NotFiniteNumberException(System.Runtime.Serialization.SerializationInfo info,  
                                        System.Runtime.Serialization.StreamingContext context);  
    // Public Instance Properties  
    public double OffendingNumber {get;}  
    // Public Instance Methods  
    public override void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // overrides Exception  
                                        System.Runtime.Serialization.StreamingContext context);  
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → ArithmeticException → NotFiniteNumberException

## NotImplementedException

class

System (mscorlib.dll)

serializable

DESCRIPTION OF TYPE GOES HERE

```
public class NotImplementedException : SystemException {  
    // Public Constructors  
    public NotImplementedException();  
    public NotImplementedException(string message);  
    public NotImplementedException(string message, Exception inner);  
    // Protected Constructors  
    protected NotImplementedException(System.Runtime.Serialization.SerializationInfo info,  
                                        System.Runtime.Serialization.StreamingContext context);  
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → NotImplementedException

## NotSupportedException

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class NotSupportedException : SystemException {  
    // Public Constructors  
    public NotSupportedException();  
    public NotSupportedException(string message);  
}
```

```

public NotSupportedException(string message, Exception innerException);
// Protected Constructors
protected NotSupportedException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → NotSupportedException

*Subclasses:* PlatformNotSupportedException

## **NullReferenceException**

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class NullReferenceException : SystemException {
// Public Constructors
public NullReferenceException();
public NullReferenceException(string message);
public NullReferenceException(string message, Exception innerException);
// Protected Constructors
protected NullReferenceException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → NullReferenceException

## **Object**

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class Object {
// Public Constructors
public Object();
// Public Static Methods
public static bool Equals(object objA, object objB);
public static bool ReferenceEquals(object objA, object objB);
// Public Instance Methods
public virtual bool Equals(object obj);
public virtual int GetHashCode();
public Type GetType();
public virtual string ToString();
// Protected Instance Methods
protected override void Finalize();
protected object MemberwiseClone();
}

```

*Subclasses:* Multiple types

*Returned By:* Multiple types

*Passed To:* Multiple types

## **ObjectDisposedException**

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class ObjectDisposedException : InvalidOperationException {  
    // Public Constructors  
    public ObjectDisposedException(string objectName);  
    public ObjectDisposedException(string objectName, string message);  
    // Protected Constructors  
    protected ObjectDisposedException(System.Runtime.Serialization.SerializationInfo info,  
                                        System.Runtime.Serialization.StreamingContext context);  
  
    // Public Instance Properties  
    public override string Message {get; } // overrides Exception  
    public string ObjectName {get; }  
    // Public Instance Methods  
    public override void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // overrides Exception  
                                        System.Runtime.Serialization.StreamingContext context);  
}
```

*Hierarchy*: Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException →  
InvalidOperationException → ObjectDisposedException

### **ObsoleteAttribute**

**sealed class**

System (mscorlib.dll)

*ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public sealed class ObsoleteAttribute : Attribute {  
    // Public Constructors  
    public ObsoleteAttribute();  
    public ObsoleteAttribute(string message);  
    public ObsoleteAttribute(string message, bool error);  
    // Public Instance Properties  
    public bool IsError {get; }  
    public string Message {get; }  
}
```

*Hierarchy*: Object → Attribute → ObsoleteAttribute

*Valid On*: Class, Struct, Enum, Constructor, Method, Property, Field, Event, Interface, Delegate

### **OutOfMemoryException**

**class**

System (mscorlib.dll)

*ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public class OutOfMemoryException : SystemException {  
    // Public Constructors  
    public OutOfMemoryException();  
    public OutOfMemoryException(string message);  
    public OutOfMemoryException(string message, Exception innerException);  
    // Protected Constructors  
    protected OutOfMemoryException(System.Runtime.Serialization.SerializationInfo info,  
                                    System.Runtime.Serialization.StreamingContext context);  
}
```

*Hierarchy*: Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException →  
OutOfMemoryException

### **OverflowException**

**class**

System (mscorlib.dll)

*ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public class OverflowException : ArithmeticException {
// Public Constructors
public OverflowException();
public OverflowException(string message);
public OverflowException(string message, Exception innerException);
// Protected Constructors
protected OverflowException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → ArithmeticException → OverflowException

**ParamArrayAttribute** sealed class  
System (mscorlib.dll) ECMA

DESCRIPTION OF TYPE GOES HERE

```
public sealed class ParamArrayAttribute : Attribute {
// Public Constructors
public ParamArrayAttribute();
}
```

*Hierarchy:* Object → Attribute → ParamArrayAttribute

*Valid On:* Parameter

**PlatformNotSupportedException** class  
System (mscorlib.dll) serializable

DESCRIPTION OF TYPE GOES HERE

```
public class PlatformNotSupportedException : NotSupportedException {
// Public Constructors
public PlatformNotSupportedException();
public PlatformNotSupportedException(string message);
public PlatformNotSupportedException(string message, Exception inner);
// Protected Constructors
protected PlatformNotSupportedException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → NotSupportedException → PlatformNotSupportedException

**Random** class  
System (mscorlib.dll) ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class Random {
// Public Constructors
public Random();
public Random(int Seed);
// Public Instance Methods
public virtual int Next();
public virtual int Next(int maxValue);
public virtual int Next(int minValue, int maxValue);
}
```

```

public virtual void NextBytes(byte[] buffer);
public virtual double NextDouble();
// Protected Instance Methods
protected virtual double Sample();
}

```

## **RankException** class

**System** (mscorlib.dll) *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```

public class RankException : SystemException {
// Public Constructors
public RankException();
public RankException(string message);
public RankException(string message, Exception innerException);
// Protected Constructors
protected RankException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy*: Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → RankException

## **ResolveEventArgs** class

**System** (mscorlib.dll)

DESCRIPTION OF TYPE GOES HERE

```

public class ResolveEventArgs : EventArgs {
// Public Constructors
public ResolveEventArgs(string name);
// Public Instance Properties
public string Name {get;}
}

```

*Hierarchy*: Object → EventArgs → ResolveEventArgs

*Passed To*: System.Reflection.ModuleResolveEventHandler.{BeginInvoke(), Invoke()},  
ResolveEventHandler.{BeginInvoke(), Invoke()}

## **ResolveEventHandler** delegate

**System** (mscorlib.dll) *serializable*

DESCRIPTION OF TYPE GOES HERE

```

public delegate Assembly ResolveEventHandler(object sender, ResolveEventArgs args);

```

*Associated Events*: AppDomain.{AssemblyResolve(), ResourceResolve(), TypeResolve()}

## **RuntimeTypeHandle** struct

**System** (mscorlib.dll) *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```

public struct RuntimeTypeHandle : System.Runtime.Serialization.ISerializable {

```

```
// Public Instance Properties
public IntPtr Value {get; }
// Public Instance Methods
public void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // implements ISerializable
                           System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → ValueType → RuntimeTypeHandle(System.Runtime.Serialization.ISerializable)

*Returned By:* ArgIterator.GetNextArgType(), Type.{GetTypeHandle(), TypeHandle}

*Passed To:* ArgIterator.GetNextArg(), Type.GetTypeFromHandle()

## SByte

**struct**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct SByte : IComparable, IFormattable, IConvertible {
// Public Static Fields
public const sbyte MaxValue; // =127
public const sbyte MinValue; // =-128
// Public Static Methods
public static sbyte Parse(string s);
public static sbyte Parse(string s, IFormatProvider provider);
public static sbyte Parse(string s, System.Globalization.NumberStyles style);
public static sbyte Parse(string s, System.Globalization.NumberStyles style, IFormatProvider provider);
// Public Instance Methods
public int CompareTo(object obj); // implements IComparable
public override bool Equals(object obj); // overrides ValueType
public override int GetHashCode(); // overrides ValueType
public TypeCode GetTypeCode(); // implements IConvertible
public override string ToString(); // overrides ValueType
public string ToString(IFormatProvider provider); // implements IConvertible
public string ToString(string format);
public string ToString(string format, IFormatProvider provider); // implements IFormattable
}
```

*Hierarchy:* Object → ValueType → SByte(IComparable, IFormattable, IConvertible)

*Returned By:* Convert.ToSByte(), Decimal.ToSByte(), IConvertible.ToSByte(),  
System.IO.BinaryReader.ReadSByte(), Math.{Abs(), Max(), Min()},  
System.Runtime.Serialization.FormatterConverter.ToSByte(),  
System.Runtime.Serialization.IFormatterConverter.ToSByte(),  
System.Runtime.Serialization.SerializationInfo.GetSByte(), System.Xml.XmlConvert.ToSByte()

*Passed To:* Multiple types

## SerializableAttribute

**sealed class**

System (mscorlib.dll)

DESCRIPTION OF TYPE GOES HERE

```
public sealed class SerializableAttribute : Attribute {
// Public Constructors
public SerializableAttribute();
}
```

*Hierarchy:* Object → Attribute → SerializableAttribute

*Valid On:* Class, Struct, Enum, Delegate

## Single

**struct**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct Single : IComparable, IFormattable, IConvertible {  
    // Public Static Fields  
    public const float Epsilon;                =1.401298E-45  
    public const float MaxValue;              =3.402823E+38  
    public const float MinValue;              =-3.402823E+38  
    public const float NaN;                    =NaN  
    public const float NegativeInfinity;      =-Infinity  
    public const float PositiveInfinity;      =Infinity  
    // Public Static Methods  
    public static bool IsInfinity(float f);  
    public static bool IsNaN(float f);  
    public static bool IsNegativeInfinity(float f);  
    public static bool IsPositiveInfinity(float f);  
    public static float Parse(string s);  
    public static float Parse(string s, IFormatProvider provider);  
    public static float Parse(string s, System.Globalization.NumberStyles style);  
    public static float Parse(string s, System.Globalization.NumberStyles style, IFormatProvider provider);  
    // Public Instance Methods  
    public int CompareTo(object value);          // implements IComparable  
    public override bool Equals(object obj);    // overrides ValueType  
    public override int GetHashCode();          // overrides ValueType  
    public TypeCode GetTypeCode();             // implements IConvertible  
    public override string ToString();          // overrides ValueType  
    public string ToString(IFormatProvider provider); // implements IConvertible  
    public string ToString(string format);  
    public string ToString(string format, IFormatProvider provider); // implements IFormattable  
}
```

*Hierarchy:* Object → ValueType → Single(IComparable, IFormattable, IConvertible)

*Returned By:* BitConverter.ToSingle(), Convert.ToSingle(), Decimal.ToSingle(), IConvertible.ToSingle(), System.IO.BinaryReader.ReadSingle(), System.Runtime.Serialization.FormatterConverter.ToSingle(), System.Runtime.Serialization.IFormatterConverter.ToSingle(), System.Runtime.Serialization.SerializationInfo.GetSingle(), System.Xml.XmlConvert.ToSingle()

*Passed To:* Multiple types

## StackOverflowException

**sealed class**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public sealed class StackOverflowException : SystemException {  
    // Public Constructors  
    public StackOverflowException();  
    public StackOverflowException(string message);  
    public StackOverflowException(string message, Exception innerException);  
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → StackOverflowException

## String

**sealed class**

DESCRIPTION OF TYPE GOES HERE

```

public sealed class String : IComparable, ICloneable, IConvertible, IEnumerable {
// Public Constructors
    public String(char *value);
    public String(char *value, int startIndex, int length);
    public String(char[] value);
    public String(char[] value, int startIndex, int length);
    public String(char c, int count);
    public String(sbyte *value);
    public String(sbyte *value, int startIndex, int length);
    public String(sbyte *value, int startIndex, int length, System.Text.Encoding enc);
// Public Static Fields
    public static readonly string Empty;
// Public Instance Properties
    public int Length {get; }
    public char this[int index]{get; }
// Public Static Methods
    public static int Compare(string strA, int indexA, string strB, int indexB, int length);
    public static int Compare(string strA, int indexA, string strB, int indexB, int length, bool ignoreCase);
    public static int Compare(string strA, int indexA, string strB, int indexB, int length, bool ignoreCase,
        System.Globalization.CultureInfo culture);
    public static int Compare(string strA, string strB);
    public static int Compare(string strA, string strB, bool ignoreCase);
    public static int Compare(string strA, string strB, bool ignoreCase, System.Globalization.CultureInfo culture);
    public static int CompareOrdinal(string strA, int indexA, string strB, int indexB, int length);
    public static int CompareOrdinal(string strA, string strB);
    public static string Concat(object arg0);
    public static string Concat(params object[] args);
    public static string Concat(object arg0, object arg1);
    public static string Concat(object arg0, object arg1, object arg2);
    public static string Concat(object arg0, object arg1, object arg2, object arg3);
    public static string Concat(params string[] values);
    public static string Concat(string str0, string str1);
    public static string Concat(string str0, string str1, string str2);
    public static string Concat(string str0, string str1, string str2, string str3);
    public static string Copy(string str);
    public static bool Equals(string a, string b);
    public static string Format(IFormatProvider provider, string format, params object[] args);
    public static string Format(string format, object arg0);
    public static string Format(string format, params object[] args);
    public static string Format(string format, object arg0, object arg1);
    public static string Format(string format, object arg0, object arg1, object arg2);
    public static string Intern(string str);
    public static string IsInterned(string str);
    public static string Join(string separator, string[] value);
    public static string Join(string separator, string[] value, int startIndex, int count);
    public static bool operator !=(string a, string b);
    public static bool operator ==(string a, string b);
// Public Instance Methods
    public object Clone(); // implements ICloneable
    public int CompareTo(object value); // implements IComparable
    public int CompareTo(string strB);
    public void CopyTo(int sourceIndex, char[] destination, int destinationIndex, int count);

```



```

public bool EndsWith(string value);
public override bool Equals(object obj); // overrides object
public bool Equals(string value);
public CharEnumerator GetEnumerator();
public override int GetHashCode(); // overrides object
public TypeCode GetTypeCode(); // implements IConvertible
public int IndexOf(char value);
public int IndexOf(char value, int startIndex);
public int IndexOf(char value, int startIndex, int count);
public int IndexOf(string value);
public int IndexOf(string value, int startIndex);
public int IndexOf(string value, int startIndex, int count);
public int IndexOfAny(char[] anyOf);
public int IndexOfAny(char[] anyOf, int startIndex);
public int IndexOfAny(char[] anyOf, int startIndex, int count);
public string Insert(int startIndex, string value);
public int LastIndexOf(char value);
public int LastIndexOf(char value, int startIndex);
public int LastIndexOf(char value, int startIndex, int count);
public int LastIndexOf(string value);
public int LastIndexOf(string value, int startIndex);
public int LastIndexOf(string value, int startIndex, int count);
public int LastIndexOfAny(char[] anyOf);
public int LastIndexOfAny(char[] anyOf, int startIndex);
public int LastIndexOfAny(char[] anyOf, int startIndex, int count);
public string PadLeft(int totalWidth);
public string PadLeft(int totalWidth, char paddingChar);
public string PadRight(int totalWidth);
public string PadRight(int totalWidth, char paddingChar);
public string Remove(int startIndex, int count);
public string Replace(char oldChar, char newChar);
public string Replace(string oldValue, string newValue);
public string[] Split(params char[] separator);
public string[] Split(char[] separator, int count);
public bool StartsWith(string value);
public string Substring(int startIndex);
public string Substring(int startIndex, int length);
public char[] ToCharArray();
public char[] ToCharArray(int startIndex, int length);
public string ToLower();
public string ToLower(System.Globalization.CultureInfo culture);
public override string ToString(); // overrides object
public string ToString(IFormatProvider provider); // implements IConvertible
public string ToUpper();
public string ToUpper(System.Globalization.CultureInfo culture);
public string Trim();
public string Trim(params char[] trimChars);
public string TrimEnd(params char[] trimChars);
public string TrimStart(params char[] trimChars);
}

```

*Returned By:* Multiple types

*Passed To:* Multiple types

## **SystemException**

**class**

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class SystemException : Exception {
// Public Constructors
public SystemException();
public SystemException(string message);
public SystemException(string message, Exception innerException);
// Protected Constructors
protected SystemException(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException

*Subclasses:* Multiple types

**ThreadStaticAttribute** class  
 System (mscorlib.dll) serializable

DESCRIPTION OF TYPE GOES HERE

```
public class ThreadStaticAttribute : Attribute {
// Public Constructors
public ThreadStaticAttribute();
}
```

*Hierarchy:* Object → Attribute → ThreadStaticAttribute

*Valid On:* Field

**TimeSpan** struct  
 System (mscorlib.dll) ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct TimeSpan : IComparable {
// Public Constructors
public TimeSpan(int hours, int minutes, int seconds);
public TimeSpan(int days, int hours, int minutes, int seconds);
public TimeSpan(int days, int hours, int minutes, int seconds, int milliseconds);
public TimeSpan(long ticks);
// Public Static Fields
public static readonly TimeSpan MaxValue; =10675199.02:48:05.4775807
public static readonly TimeSpan MinValue; =-10675199.02:48:05.4775808
public const long TicksPerDay; =864000000000
public const long TicksPerHour; =36000000000
public const long TicksPerMillisecond; =10000
public const long TicksPerMinute; =600000000
public const long TicksPerSecond; =10000000
public static readonly TimeSpan Zero; =00:00:00
// Public Instance Properties
public int Days {get; }
public int Hours {get; }
public int Milliseconds {get; }
public int Minutes {get; }
public int Seconds {get; }
public long Ticks {get; }
public double TotalDays {get; }
public double TotalHours {get; }
```

```

public double TotalMilliseconds {get; }
public double TotalMinutes {get; }
public double TotalSeconds {get; }
// Public Static Methods
public static int Compare(TimeSpan t1, TimeSpan t2);
public static bool Equals(TimeSpan t1, TimeSpan t2);
public static TimeSpan FromDays(double value);
public static TimeSpan FromHours(double value);
public static TimeSpan FromMilliseconds(double value);
public static TimeSpan FromMinutes(double value);
public static TimeSpan FromSeconds(double value);
public static TimeSpan FromTicks(long value);
public static TimeSpan Parse(string s);
public static TimeSpan operator -(TimeSpan t);
public static TimeSpan operator -(TimeSpan t1, TimeSpan t2);
public static TimeSpan operator +(TimeSpan t);
public static TimeSpan operator +(TimeSpan t1, TimeSpan t2);
public static bool operator !=(TimeSpan t1, TimeSpan t2);
public static bool operator <(TimeSpan t1, TimeSpan t2);
public static bool operator <=(TimeSpan t1, TimeSpan t2);
public static bool operator ==(TimeSpan t1, TimeSpan t2);
public static bool operator >(TimeSpan t1, TimeSpan t2);
public static bool operator >=(TimeSpan t1, TimeSpan t2);
// Public Instance Methods
public TimeSpan Add(TimeSpan ts);
public int CompareTo(object value); // implements IComparable
public TimeSpan Duration();
public override bool Equals(object value); // overrides ValueType
public override int GetHashCode(); // overrides ValueType
public TimeSpan Negate();
public TimeSpan Subtract(TimeSpan ts);
public override string ToString(); // overrides ValueType
}

```

*Hierarchy:* Object → ValueType → TimeSpan(IComparable)

*Returned By:* DateTime.{Subtract(), TimeOfDay}, System.Globalization.DaylightTime.Delta, TimeZone.GetUtcOffset(), System.Xml.XmlConvert.ToTimeSpan()

*Passed To:* Multiple types

## TimeZone

**abstract class**

System (mscorlib.dll)

*serializable*

DESCRIPTION OF TYPE GOES HERE

```

public abstract class TimeZone {
// Protected Constructors
protected TimeZone();
// Public Static Properties
public static TimeZone CurrentTimeZone {get; }
// Public Instance Properties
public abstract string DaylightName {get; }
public abstract string StandardName {get; }
// Public Static Methods
public static bool IsDaylightSavingTime(DateTime time, System.Globalization.DaylightTime daylightTimes);
// Public Instance Methods
}

```

```

public abstract DaylightTime GetDaylightChanges(int year);
public abstract TimeSpan GetUtcOffset(DateTime time);
public virtual bool IsDaylightSavingTime(DateTime time);
public virtual DateTime ToLocalTime(DateTime time);
public virtual DateTime ToUniversalTime(DateTime time);
}

```

**Type** **abstract class**  
**System (mscorlib.dll)** **ECMA, serializable**

DESCRIPTION OF TYPE GOES HERE

```

public abstract class Type : System.Reflection.MemberInfo , System.Reflection.IReflect {
// Protected Constructors
protected Type();
// Public Static Fields
public static readonly char Delimiter; //0x0000002E
public static readonly Type[] EmptyTypes; //System.Type[]
public static readonly MemberFilter FilterAttribute; //System.Reflection.MemberFilter
public static readonly MemberFilter FilterName; //System.Reflection.MemberFilter
public static readonly MemberFilter FilterNameIgnoreCase; //System.Reflection.MemberFilter
public static readonly object Missing; //System.Reflection.Missing
// Public Static Properties
public static Binder DefaultBinder {get; }
// Public Instance Properties
public abstract Assembly Assembly {get; }
public abstract string AssemblyQualifiedName {get; }
public TypeAttributes Attributes {get; }
public abstract Type BaseType {get; }
public override Type DeclaringType {get; } // overrides System.Reflection.MemberInfo
public abstract string FullName {get; }
public abstract Guid GUID {get; }
public bool HasElementType {get; }
public bool IsAbstract {get; }
public bool IsAnsiClass {get; }
public bool IsArray {get; }
public bool IsAutoClass {get; }
public bool IsAutoLayout {get; }
public bool IsByRef {get; }
public bool IsClass {get; }
public bool IsCOMObject {get; }
public bool IsContextful {get; }
public bool IsEnum {get; }
public bool IsExplicitLayout {get; }
public bool IsImport {get; }
public bool IsInterface {get; }
public bool IsLayoutSequential {get; }
public bool IsMarshalByRef {get; }
public bool IsNestedAssembly {get; }
public bool IsNestedFamANDAssem {get; }
public bool IsNestedFamily {get; }
public bool IsNestedFamORAssem {get; }
public bool IsNestedPrivate {get; }
public bool IsNestedPublic {get; }
public bool IsNotPublic {get; }
public bool IsPointer {get; }

```

```

public bool IsPrimitive {get; }
public bool IsPublic {get; }
public bool IsSealed {get; }
public bool IsSerializable {get; }
public bool IsSpecialName {get; }
public bool IsUnicodeClass {get; }
public bool IsValueType {get; }
public override MemberTypes MemberType {get; } // overrides System.Reflection.MemberInfo
public abstract Module Module {get; }
public abstract string Namespace {get; }
public override Type ReflectedType {get; } // overrides System.Reflection.MemberInfo
public abstract RuntimeTypeHandle TypeHandle {get; }
public ConstructorInfo TypeInitializer {get; }
public abstract Type UnderlyingSystemType {get; } // implements System.Reflection.IReflect
// Public Static Methods
public static Type GetType(string typeName);
public static Type GetType(string typeName, bool throwOnError);
public static Type GetType(string typeName, bool throwOnError, bool ignoreCase);
public static Type[] GetTypeArray(object[] args);
public static TypeCode GetTypeCode(Type type);
public static Type GetTypeFromHandle(RuntimeTypeHandle handle);
public static RuntimeTypeHandle GetTypeHandle(object o);
// Public Instance Methods
public override bool Equals(object o); // overrides object
public bool Equals(Type o);
public virtual Type[] FindInterfaces(System.Reflection.TypeFilter filter, object filterCriteria);
public virtual MemberInfo[] FindMembers(System.Reflection.MemberTypes memberType,
System.Reflection.BindingFlags bindingAttr,
System.Reflection.MemberFilter filter, object filterCriteria);
public virtual int GetArrayRank();
public ConstructorInfo GetConstructor(System.Reflection.BindingFlags bindingAttr,
System.Reflection.Binder binder,
System.Reflection.CallingConventions callConvention, Type[] types,
System.Reflection.ParameterModifier[] modifiers);
public ConstructorInfo GetConstructor(System.Reflection.BindingFlags bindingAttr,
System.Reflection.Binder binder, Type[] types,
System.Reflection.ParameterModifier[] modifiers);
public ConstructorInfo GetConstructor(Type[] types);
public ConstructorInfo[] GetConstructors();
public abstract ConstructorInfo[] GetConstructors(System.Reflection.BindingFlags bindingAttr);
public virtual MemberInfo[] GetDefaultMembers();
public abstract Type GetElementType();
public EventInfo GetEvent(string name);
public abstract EventInfo GetEvent(string name, System.Reflection.BindingFlags bindingAttr);
public virtual EventInfo[] GetEvents();
public abstract EventInfo[] GetEvents(System.Reflection.BindingFlags bindingAttr);
public FieldInfo GetField(string name);
public abstract FieldInfo GetField(string name, // implements System.Reflection.IReflect
System.Reflection.BindingFlags bindingAttr);
public FieldInfo[] GetFields();
public abstract FieldInfo[] GetFields( // implements System.Reflection.IReflect
System.Reflection.BindingFlags bindingAttr);
public override int GetHashCode(); // overrides object
public Type GetInterface(string name);
public abstract Type GetInterface(string name, bool ignoreCase);

```

```

public virtual InterfaceMapping GetInterfaceMap(Type interfaceType);
public abstract Type[] GetInterfaces();
public MemberInfo[] GetMember(string name);
public virtual MemberInfo[] GetMember(string name, // implements System.Reflection.IReflect
                                         System.Reflection.BindingFlags bindingAttr);
public virtual MemberInfo[] GetMember(string name, System.Reflection.MemberTypes type,
                                         System.Reflection.BindingFlags bindingAttr);
public MemberInfo[] GetMembers();
public abstract MemberInfo[] GetMembers( // implements System.Reflection.IReflect
                                         System.Reflection.BindingFlags bindingAttr);
public MethodInfo GetMethod(string name);
public MethodInfo GetMethod(string name, // implements System.Reflection.IReflect
                              System.Reflection.BindingFlags bindingAttr);
public MethodInfo GetMethod(string name, System.Reflection.BindingFlags bindingAttr,
                              System.Reflection.Binder binder,
                              System.Reflection.CallingConventions callConvention, Type[] types,
                              System.Reflection.ParameterModifier[] modifiers);
public MethodInfo GetMethod(string name, // implements System.Reflection.IReflect
                              System.Reflection.BindingFlags bindingAttr,
                              System.Reflection.Binder binder,
                              Type[] types,
                              System.Reflection.ParameterModifier[] modifiers);
public MethodInfo GetMethod(string name, Type[] types);
public MethodInfo GetMethod(string name, Type[] types, System.Reflection.ParameterModifier[] modifiers);
public MethodInfo[] GetMethods();
public abstract MethodInfo[] GetMethods( // implements System.Reflection.IReflect
                                         System.Reflection.BindingFlags bindingAttr);
public Type GetNestedType(string name);
public abstract Type GetNestedType(string name, System.Reflection.BindingFlags bindingAttr);
public Type[] GetNestedTypes();
public abstract Type[] GetNestedTypes(System.Reflection.BindingFlags bindingAttr);
public PropertyInfo[] GetProperties();
public abstract PropertyInfo[] GetProperties( // implements System.Reflection.IReflect
                                              System.Reflection.BindingFlags bindingAttr);
public PropertyInfo GetProperty(string name);
public PropertyInfo GetProperty(string name, // implements System.Reflection.IReflect
                                 System.Reflection.BindingFlags bindingAttr);
public PropertyInfo GetProperty(string name, // implements System.Reflection.IReflect
                                 System.Reflection.BindingFlags bindingAttr,
                                 System.Reflection.Binder binder,
                                 Type returnType, Type[] types,
                                 System.Reflection.ParameterModifier[] modifiers);
public PropertyInfo GetProperty(string name, Type returnType);
public PropertyInfo GetProperty(string name, Type[] types);
public PropertyInfo GetProperty(string name, Type returnType, Type[] types);
public PropertyInfo GetProperty(string name, Type returnType, Type[] types,
                                 System.Reflection.ParameterModifier[] modifiers);
public object InvokeMember(string name, System.Reflection.BindingFlags invokeAttr,
                             System.Reflection.Binder binder, object target, object[] args);
public object InvokeMember(string name, System.Reflection.BindingFlags invokeAttr,
                             System.Reflection.Binder binder, object target, object[] args,
                             System.Globalization.CultureInfo culture);
public abstract object InvokeMember(string name, // implements System.Reflection.IReflect
                                     System.Reflection.BindingFlags invokeAttr,
                                     System.Reflection.Binder binder,
                                     object target, object[] args,

```

```

        System.Reflection.ParameterModifier[] modifiers,
        System.Globalization.CultureInfo culture,
        string[] namedParameters);
public virtual bool IsAssignableFrom(Type c);
public virtual bool IsInstanceOfType(object o);
public virtual bool IsSubclassOf(Type c);
public override string Tostring(); // overrides object
// Protected Instance Methods
protected abstract TypeAttributes GetAttributeFlagsImpl();
protected abstract ConstructorInfo GetConstructorImpl(System.Reflection.BindingFlags bindingAttr,
        System.Reflection.Binder binder,
        System.Reflection.CallingConventions callConvention,
        Type[] types,
        System.Reflection.ParameterModifier[] modifiers);
protected abstract MethodInfo GetMethodImpl(string name, System.Reflection.BindingFlags bindingAttr,
        System.Reflection.Binder binder,
        System.Reflection.CallingConventions callConvention, Type[] types,
        System.Reflection.ParameterModifier[] modifiers);
protected abstract PropertyInfo GetPropertyImpl(string name, System.Reflection.BindingFlags bindingAttr,
        System.Reflection.Binder binder, Type returnType, Type[] types,
        System.Reflection.ParameterModifier[] modifiers);
protected abstract bool HasElementTypeImpl();
protected abstract bool isArrayImpl();
protected abstract bool IsByRefImpl();
protected abstract bool IsCOMObjectImpl();
protected virtual bool IsContextfulImpl();
protected virtual bool IsMarshalByRefImpl();
protected abstract bool IsPointerImpl();
protected abstract bool IsPrimitiveImpl();
protected virtual bool IsValueTypeImpl();
}

```

*Hierarchy*: Object → System.Reflection.MemberInfo(System.Reflection.ICustomAttributeProvider) → Type(System.Reflection.IReflect)

*Subclasses*: System.Reflection.TypeDelegator, System.Reflection.Emit.{EnumBuilder, TypeBuilder}

*Returned By*: Multiple types

*Passed To*: Multiple types

## TypeCode

enum

System (mscorlib.dll)

serializable

DESCRIPTION OF TYPE GOES HERE

```

public enum TypeCode {
    Empty = 0,
    Object = 1,
    DBNull = 2,
    Boolean = 3,
    Char = 4,
    SByte = 5,
    Byte = 6,
    Int16 = 7,
    UInt16 = 8,
    Int32 = 9,
    UInt32 = 10,
    Int64 = 11,
}

```

```

    UInt64 = 12,
    Single = 13,
    Double = 14,
    Decimal = 15,
    DateTime = 16,
    String = 18
}

```

*Hierarchy:* Object → ValueType → Enum(IComparable, IFormattable, IConvertible) → TypeCode

*Returned By:* Multiple types

*Passed To:* Convert.ChangeType(), System.Runtime.Serialization.FormatterConverter.Convert(), System.Runtime.Serialization.IFormatterConverter.Convert()

## TypeInitializationException

sealed class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public sealed class TypeInitializationException : SystemException {
// Public Constructors
    public TypeInitializationException(string fullTypeName, Exception innerException);
// Public Instance Properties
    public string TypeName {get; }
// Public Instance Methods
    public override void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // overrides Exception
                                     System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → TypeInitializationException

## TypeLoadException

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class TypeLoadException : SystemException {
// Public Constructors
    public TypeLoadException();
    public TypeLoadException(string message);
    public TypeLoadException(string message, Exception inner);
// Protected Constructors
    protected TypeLoadException(System.Runtime.Serialization.SerializationInfo info,
                                 System.Runtime.Serialization.StreamingContext context);
// Public Instance Properties
    public override string Message {get; } // overrides Exception
    public string TypeName {get; }
// Public Instance Methods
    public override void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // overrides Exception
                                     System.Runtime.Serialization.StreamingContext context);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → TypeLoadException

*Subclasses:* DllNotFoundException, EntryPointNotFoundException

## TypeUnloadedException

class

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class TypeUnloadedException : SystemException {  
    // Public Constructors  
    public TypeUnloadedException();  
    public TypeUnloadedException(string message);  
    public TypeUnloadedException(string message, Exception innerException);  
    // Protected Constructors  
    protected TypeUnloadedException(System.Runtime.Serialization.SerializationInfo info,  
        System.Runtime.Serialization.StreamingContext context);  
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → TypeUnloadedException

## UInt16

struct

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct UInt16 : IComparable, IFormattable, IConvertible {  
    // Public Static Fields  
    public const ushort MaxValue; // =65535  
    public const ushort MinValue; // =0  
    // Public Static Methods  
    public static ushort Parse(string s);  
    public static ushort Parse(string s, IFormatProvider provider);  
    public static ushort Parse(string s, System.Globalization.NumberStyles style);  
    public static ushort Parse(string s, System.Globalization.NumberStyles style, IFormatProvider provider);  
    // Public Instance Methods  
    public int CompareTo(object value); // implements IComparable  
    public override bool Equals(object obj); // overrides ValueType  
    public override int GetHashCode(); // overrides ValueType  
    public TypeCode GetTypeCode(); // implements IConvertible  
    public override string ToString(); // overrides ValueType  
    public string ToString(IFormatProvider provider); // implements IConvertible  
    public string ToString(string format);  
    public string ToString(string format, IFormatProvider provider); // implements IFormattable  
}
```

*Hierarchy:* Object → ValueType → UInt16(IComparable, IFormattable, IConvertible)

*Returned By:* BitConverter.ToUInt16(), Convert.ToUInt16(), Decimal.ToUInt16(), IConvertible.ToUInt16(), System.IO.BinaryReader.ReadUInt16(), System.Runtime.Serialization.FormatterConverter.ToUInt16(), System.Runtime.Serialization.IFormatterConverter.ToUInt16(), System.Runtime.Serialization.SerializationInfo.GetUInt16(), System.Xml.XmlConvert.ToUInt16()

*Passed To:* Multiple types

## UInt32

struct

System (mscorlib.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public struct UInt32 : IComparable, IFormattable, IConvertible {  
    // Public Static Fields
```

```

public const uint MaxValue;           =4294967295
public const uint MinValue;           =0
// Public Static Methods
public static uint Parse(string s);
public static uint Parse(string s, IFormatProvider provider);
public static uint Parse(string s, System.Globalization.NumberStyles style);
public static uint Parse(string s, System.Globalization.NumberStyles style, IFormatProvider provider);
// Public Instance Methods
public int CompareTo(object value);           // implements IComparable
public override bool Equals(object obj);     // overrides ValueType
public override int GetHashCode();          // overrides ValueType
public TypeCode GetTypeCode();             // implements IConvertible
public override string ToString();          // overrides ValueType
public string ToString(IFormatProvider provider); // implements IConvertible
public string ToString(string format);
public string ToString(string format, IFormatProvider provider); // implements IFormattable
}

```

*Hierarchy:* Object → ValueType → UInt32(IComparable, IFormattable, IConvertible)

*Returned By:* BitConverter.ToUInt32(), Convert.ToUInt32(), Decimal.ToUInt32(), IConvertible.ToUInt32(), System.IO.BinaryReader.ReadUInt32(), System.Reflection.AssemblyAlgorithmIdAttribute.AlgorithmId, System.Reflection.AssemblyFlagsAttribute.Flags, System.Runtime.Serialization.FormatterConverter.ToUInt32(), System.Runtime.Serialization.IFormatterConverter.ToUInt32(), System.Runtime.Serialization.SerializationInfo.GetUInt32(), UIntPtr.ToUInt32(), System.Xml.XmlConvert.ToUInt32()

*Passed To:* Multiple types

## UInt64

**struct**

**System (mscorlib.dll)**

*ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```

public struct UInt64 : IComparable, IFormattable, IConvertible {
// Public Static Fields
public const ulong MaxValue;           =18446744073709551615
public const ulong MinValue;           =0
// Public Static Methods
public static ulong Parse(string s);
public static ulong Parse(string s, IFormatProvider provider);
public static ulong Parse(string s, System.Globalization.NumberStyles style);
public static ulong Parse(string s, System.Globalization.NumberStyles style, IFormatProvider provider);
// Public Instance Methods
public int CompareTo(object value);           // implements IComparable
public override bool Equals(object obj);     // overrides ValueType
public override int GetHashCode();          // overrides ValueType
public TypeCode GetTypeCode();             // implements IConvertible
public override string ToString();          // overrides ValueType
public string ToString(IFormatProvider provider); // implements IConvertible
public string ToString(string format);
public string ToString(string format, IFormatProvider provider); // implements IFormattable
}

```

*Hierarchy:* Object → ValueType → UInt64(IComparable, IFormattable, IConvertible)

*Returned By:* BitConverter.ToUInt64(), Convert.ToUInt64(), Decimal.ToUInt64(),  
 IConvertible.ToUInt64(), System.IO.BinaryReader.ReadUInt64(),  
 System.IO.IsolatedStorage.IsolatedStorage.{CurrentSize, MaximumSize},  
 System.Runtime.Serialization.FormatterConverter.ToUInt64(),  
 System.Runtime.Serialization.IFormatterConverter.ToUInt64(),  
 System.Runtime.Serialization.SerializationInfo.GetUInt64(), UIntPtr.ToUInt64(),  
 System.Xml.XmlConvert.ToUInt64()

*Passed To:* Multiple types

**UIntPtr** **struct**  
 System (mscorlib.dll) *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public struct UIntPtr : System.Runtime.Serialization.ISerializable {
// Public Constructors
    public UIntPtr(uint value);
    public UIntPtr(ulong value);
    public UIntPtr(void *value);
// Public Static Fields
    public static readonly UIntPtr Zero; =0
// Public Static Properties
    public static int Size {get; }
// Public Static Methods
    public static bool operator !=(UIntPtr value1, UIntPtr value2);
    public static bool operator ==(UIntPtr value1, UIntPtr value2);
    public static explicit operator uint(UIntPtr value);
    public static explicit operator ulong(UIntPtr value);
    public static explicit operator UIntPtr(uint value);
    public static explicit operator UIntPtr(ulong value);
    public static explicit operator UIntPtr(void *value);
    public static explicit operator Void(UIntPtr value);
// Public Instance Methods
    public override bool Equals(object obj); // overrides ValueType
    public override int GetHashCode(); // overrides ValueType
    public void* ToPointer();
    public override string ToString(); // overrides ValueType
    public uint ToUInt32();
    public ulong ToUInt64();
}
```

*Hierarchy:* Object → ValueType → UIntPtr(System.Runtime.Serialization.ISerializable)

**UnauthorizedAccessException** **class**  
 System (mscorlib.dll) *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public class UnauthorizedAccessException : SystemException {
// Public Constructors
    public UnauthorizedAccessException();
    public UnauthorizedAccessException(string message);
    public UnauthorizedAccessException(string message, Exception inner);
// Protected Constructors
    protected UnauthorizedAccessException(System.Runtime.Serialization.SerializationInfo info,
        System.Runtime.Serialization.StreamingContext context);
}
```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → UnauthorizedAccessException

**UnhandledExceptionEventArgs** class

System (mscorlib.dll) ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public class UnhandledExceptionEventArgs : EventArgs {
// Public Constructors
    public UnhandledExceptionEventArgs(object exception, bool isTerminating);
// Public Instance Properties
    public object ExceptionObject {get; }
    public bool IsTerminating {get; }
}
```

*Hierarchy:* Object → EventArgs → UnhandledExceptionEventArgs

*Passed To:* UnhandledExceptionHandler.{BeginInvoke(), Invoke()}

**UnhandledExceptionHandler** delegate

System (mscorlib.dll) ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```
public delegate void UnhandledExceptionHandler(object sender, UnhandledExceptionEventArgs e);
```

*Associated Events:* AppDomain.UnhandledException()

**Uri** class

System (system.dll) ECMA, serializable, marshal by reference

DESCRIPTION OF TYPE GOES HERE

```
public class Uri : MarshalByRefObject, System.Runtime.Serialization.ISerializable {
// Public Constructors
    public Uri(string uriString);
    public Uri(string uriString, bool dontEscape);
    public Uri(Uri baseUri, string relativeUri);
    public Uri(Uri baseUri, string relativeUri, bool dontEscape);
// Protected Constructors
    protected Uri(System.Runtime.Serialization.SerializationInfo serializationInfo,
        System.Runtime.Serialization.StreamingContext streamingContext);
// Public Static Fields
    public static readonly string SchemeDelimiter;           =://
    public static readonly string UriSchemeFile;           =file
    public static readonly string UriSchemeFtp;             =ftp
    public static readonly string UriSchemeGopher;         =gopher
    public static readonly string UriSchemeHttp;           =http
    public static readonly string UriSchemeHttps;          =https
    public static readonly string UriSchemeMailto;         =mailto
    public static readonly string UriSchemeNews;           =news
    public static readonly string UriSchemeNntp;           =nntp
// Public Instance Properties
    public string AbsolutePath {get; }
    public string AbsoluteUri {get; }
    public string Authority {get; }
```

```

public string Fragment {get; }
public string Host {get; }
public UriHostNameType HostNameType {get; }
public bool IsDefaultPort {get; }
public bool IsFile {get; }
public bool IsLoopback {get; }
public bool IsUnc {get; }
public string LocalPath {get; }
public string PathAndQuery {get; }
public int Port {get; }
public string Query {get; }
public string Scheme {get; }
public string[] Segments {get; }
public bool UserEscaped {get; }
public string UserInfo {get; }
// Public Static Methods
public static UriHostNameType CheckHostName(string name);
public static bool CheckSchemeName(string schemeName);
public static int FromHex(char digit);
public static string HexEscape(char character);
public static char HexUnescape(string pattern, ref int index);
public static bool IsHexDigit(char character);
public static bool IsHexEncoding(string pattern, int index);
// Protected Static Methods
protected static string EscapeString(string str);
protected static bool IsExcludedCharacter(char character);
// Public Instance Methods
public override bool Equals(object comparand); // overrides object
public override int GetHashCode(); // overrides object
public string GetLeftPart(UriPartial part);
public string MakeRelative(Uri toUri);
public override string ToString(); // overrides object
// Protected Instance Methods
protected virtual void Canonicalize();
protected virtual void CheckSecurity();
protected virtual void Escape();
protected virtual bool IsBadFileSystemCharacter(char character);
protected virtual bool IsReservedCharacter(char character);
protected virtual void Parse();
protected virtual string Unescape(string path);
}

```

*Hierarchy:* Object → MarshalByRefObject → Uri(System.Runtime.Serialization.ISerializable)

*Returned By:* System.Net.Cookie.CommentUri, System.Net.HttpWebRequest.Address, System.Net.IWebProxy.GetProxy(), System.Net.ServicePoint.Address, System.Net.WebProxy.{Address, GetProxy()}, System.Net.WebRequest.RequestUri, System.Net.WebResponse.ResponseUri, UriBuilder.Uri, System.Xml.XmlResolver.ResolveUri()

*Passed To:* Multiple types

|                                  |              |
|----------------------------------|--------------|
| <b>UriBuilder</b>                | <b>class</b> |
| System (system.dll)              | <b>ECMA</b>  |
| DESCRIPTION OF TYPE GOES HERE    |              |
| public class <b>UriBuilder</b> { |              |

```

// Public Constructors
public UriBuilder();
public UriBuilder(string uri);
public UriBuilder(string schemeName, string hostName);
public UriBuilder(string scheme, string host, int portNumber);
public UriBuilder(string scheme, string host, int port, string pathValue);
public UriBuilder(string scheme, string host, int port, string path, string extraValue);
public UriBuilder(Uri uri);
// Public Instance Properties
public string Fragment {set; get; }
public string Host {set; get; }
public string Password {set; get; }
public string Path {set; get; }
public int Port {set; get; }
public string Query {set; get; }
public string Scheme {set; get; }
public Uri Uri {get; }
public string UserName {set; get; }
// Public Instance Methods
public override bool Equals(object rparam); // overrides object
public override int GetHashCode(); // overrides object
public override string ToString(); // overrides object
}

```

**UriFormatException****class**

System (system.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public class UriFormatException : FormatException {
// Public Constructors
public UriFormatException();
public UriFormatException(string textString);
// Protected Constructors
protected UriFormatException(System.Runtime.Serialization.SerializationInfo serializationInfo,
System.Runtime.Serialization.StreamingContext streamingContext);
}

```

*Hierarchy:* Object → Exception(System.Runtime.Serialization.ISerializable) → SystemException → FormatException → UriFormatException

**UriHostNameType****enum**

System (system.dll)

ECMA, serializable

DESCRIPTION OF TYPE GOES HERE

```

public enum UriHostNameType {
    Unknown = 0,
    Basic = 1,
    Dns = 2,
    IPv4 = 3,
    IPv6 = 4
}

```

*Hierarchy:* Object → ValueType → Enum(IComparable, IFormattable, IConvertible) → UriHostNameType

*Returned By:* Uri.{CheckHostName(), HostNameType}

**UriPartial** **enum**  
System (system.dll) *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public enum UriPartial {  
    Scheme = 0,  
    Authority = 1,  
    Path = 2  
}
```

*Hierarchy:* Object → ValueType → Enum(IComparable, IFormattable, IConvertible) → UriPartial

*Passed To:* Uri.GetLeftPart()

**ValueType** **abstract class**  
System (mscorlib.dll) *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public abstract class ValueType {  
    // Protected Constructors  
    protected ValueType();  
    // Public Instance Methods  
    public override bool Equals(object obj); // overrides object  
    public override int GetHashCode(); // overrides object  
    public override string ToString(); // overrides object  
}
```

*Subclasses:* Multiple types

**Version** **sealed class**  
System (mscorlib.dll) *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```
public sealed class Version : ICloneable, IComparable {  
    // Public Constructors  
    public Version();  
    public Version(int major, int minor);  
    public Version(int major, int minor, int build);  
    public Version(int major, int minor, int build, int revision);  
    public Version(string version);  
    // Public Instance Properties  
    public int Build {get;}  
    public int Major {get;}  
    public int Minor {get;}  
    public int Revision {get;}  
    // Public Static Methods  
    public static bool operator !=(Version v1, Version v2);  
    public static bool operator <(Version v1, Version v2);  
    public static bool operator <=(Version v1, Version v2);  
    public static bool operator ==(Version v1, Version v2);  
    public static bool operator >(Version v1, Version v2);  
    public static bool operator >=(Version v1, Version v2);  
    // Public Instance Methods
```

```

public object Clone(); // implements ICloneable
public int CompareTo(object version); // implements IComparable
public override bool Equals(object obj); // overrides object
public override int GetHashCode(); // overrides object
public override string ToString(); // overrides object
public string ToString(int fieldCount);
}

```

*Returned By:* Environment.Version, System.Net.HttpWebRequest.ProtocolVersion, System.Net.HttpWebResponse.ProtocolVersion, System.Net.ServicePoint.ProtocolVersion, System.Reflection.AssemblyName.Version

*Passed To:* System.Net.HttpWebRequest.ProtocolVersion, System.Reflection.Assembly.GetSatelliteAssembly(), System.Reflection.AssemblyName.Version

**Void** **struct**  
**System (mscorlib.dll)** *ECMA, serializable*

DESCRIPTION OF TYPE GOES HERE

```

public struct Void {
// No public or protected members
}

```

*Hierarchy:* Object → ValueType → Void

*Returned By:* Multiple types

*Passed To:* ArgIterator.ArgIterator(), IntPtr.IntPtr(), System.Reflection.Pointer.Box(), UIntPtr.UIntPtr()

**WeakReference** **class**  
**System (mscorlib.dll)** *serializable*

DESCRIPTION OF TYPE GOES HERE

```

public class WeakReference : System.Runtime.Serialization.ISerializable {
// Public Constructors
public WeakReference(object target);
public WeakReference(object target, bool trackResurrection);
// Protected Constructors
protected WeakReference(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context);
// Public Instance Properties
public virtual bool IsAlive {get; }
public virtual object Target {set; get; }
public virtual bool TrackResurrection {get; }
// Public Instance Methods
public virtual void GetObjectData(System.Runtime.Serialization.SerializationInfo info, // implements ISerializable
System.Runtime.Serialization.StreamingContext context);
// Protected Instance Methods
protected override void Finalize(); // overrides object
}

```

*Passed To:* GC.GetGeneration()