Overview of .NET Class Libraries

The .NET Framework provides a large number of classes, interfaces and data types that you can use to develop Windows applications and services.

In general, these libraries have been designed to replace older technologies, including the Microsoft Foundation Classes (MFC) and Active Template Library (ALT) for Visual C++, the Component Object Model (COM) and the ActiveX model used by all languages, and also the Win32 API which is the core of Windows programming concepts.

While the .NET Framework must obviously still use various technologies like the Win32 API and COM, it hides the complex details away from the programmer and can make developing applications for Windows a much easier task. In addition, since all .NET languages use the exact same set of classes, interfaces and data types, moving from one .NET language to another is much simpler.

Each area of the .NET Framework includes many terms that may be a bit new, including classes, interfaces, enumerations, events, delegates, namespaces and assemblies. Here is a brief explanation of the terms.

Namespace

A namespace is a collection of classes, enumerations and delegates under a single name. This provides organization to the entire framework and also allows the name name to be used again in different namespaces, without conflicts. The same concept is used in C++ namespaces, Java packages and XML namespaces. Generally, a .NET namespace is contained within a single Assembly (.DLL or .EXE).

Classes

Classes are code that implements a set of features and can be used directly by the developer. In many cases a class is an implementation of one or more interfaces. Classes in the .NET Framework also support the use of inheritance, so one class can derive from another and add (or override) features from the base class.

Enumerations

An enumeration is a simple list of constant values with names assigned.

Generally it is easier for programmers to remember symbolic names than to remember a long list of number values.

Events

An event is triggered wheneverthe .NET Framework has detected a change in status that you may be interested in. Events are similar to standard Windows messages.

Delegates

A delegate is a special class used to implement callbacks into your source code. Basically, you are declaring that when some event occurs, you would like the .NET Framework to call a method in your application. The handle to the method will be stored in a delegate class, which can possible have several methods listed which can be called in sequence. Delegate classes can be used for traditional event notifications as well as callbacks.

Assembly

This term is used instead of application. Basically an assembly is made up of one or more compiled source code units that are grouped together into a single unit. Assemblies can have either a .DLL or .EXE extension. In general, a .DLL assembly provides classes and services that can be used by other assemblies, while .EXE assemblies areequivalent to an application and will make use of other assemblies. Each assembly file also includes an assembly manifest, which is meta information that describes the assembly's name, version and dependencies to other assemblies. The meta information is a large part of ability of the .NET Framework to avoid the symptom sometimes called DLL hell. The .NET Framework installs its assemblies under the Windows system folder. Applications are free to use assemblies from this global catalog, or to install assemblies in their own application folder. In that case, .NET will use the application provided assemblies instead of the system assemblies. This allows you to ship versions of assemblies that you know work with your application and avoid problems when users upgrade the system assemblies.

Major Namespaces in the .NET Framework

Microsoft.CSharp

Contains classes that support compilation and code generation using the C# language.

Microsoft.JScript

Contains the JScript runtime and classes that support compilation and code generation using the JScript language.

Microsoft.VisualBasic

Contains the Visual Basic .NET runtime. This runtime is used with the Visual Basic .NET language. This namespace also contains classes that support compilation and code generation using the Visual Basic .NET language.

Microsoft.Vsa

Contains interfaces that allow you to integrate script for the .NET Framework script engines into applications, and to compile and execute code at run time.

Interface	***Description***
IVsaCode	Represents code to be compiled by a script engine.
IVsaEngine	Defines the methods and properties supported by script engines.
IVsaError	Represents an error detected by the engine during the call to the IVsaEngine.Compile method.
IVsaGlobalItem	Describes global objects added to the script engine.
IVsaItem	Defines the methods and properties all script engine items are required to support.
IVsaItems	Defines a collection of items that can be accessed using an index or a name.
IVsaPersistSite	Manages a project's code using save and load operations.
IVsaReferenceItem	Describes references added to the script engine.
IVsaSite	Defines the interface the host (developer's environment) must support to communicate with the script engine.

Enumeration	***Description***
VsaError	The exceptions that can be raises by the script engine.
VsaItemFlag	Identifiers for script objects, including Class, Module or None.
VsaItemType	Identifiers for script objects, including Code, Reference and AppGlobal.

Microsoft.Win32

Provides two types of classes: those that handle events raised by the operating system and those that manipulate the system registry.

Classes	***Description***
PowerModeChangedEventArgs	Holds information about a PowerModeChanged event.
Registry	Base class that provides access to the Windows registry keys and values.
RegistryKey	Class that represents a single key in the registry.
SessionEndedEventArgs	Holds information about a SessionEnded event.
SessionEndingEventArgs	Holds information about a SessionEnding event.
SystemEvents	Global set of system events. Cannot be inherited from.
TimerElapsedEventArgs	Holds information about a TimerElapsed event.
UserPreferenceChangedEventArgs	Holds information about a UserPreferenceChanged event.
UserPreferenceChangingEventArgs	Holds information about a UserPreferenceChanging event.

Delegate	***Description***
PowerModeChangedEventHandler	Represents the method that will be called for PowerModeChanged events.

Delegate	***Description***
SessionEndedEventHandler	Represents the method that will be called for SessionEnded events.
SessionEndingEventHandler	Represents the method that will be called for SessionEnding events.
TimerElapsedEventHandler	Represents the method that will be called for TimerElapsed events.
UserPreferenceChangedEventHandler	Represents the method that will be called for UserPreferenceChanged events.
UserPreferenceChangingEventHandler	Represents the method that will be called for UserPreferenceChanging events.

Enumeration	***Description***
PowerModes	List of supported power modes.
RegistryHive	List of top-level hives for foreign machines.
SessionEndReasons	List of reasons for ending a session.
UserPreferenceCategory	Identifies what area of user preference was changed.

System

Contains fundamental classes and base classes that define commonly used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.

Other classes provide services supporting data type conversion, method parameter manipulation, mathematics, remote and local program invocation, application environment management, and supervision of managed and unmanaged applications.

Class*	***Description***
Activator	Used to create local or remote objects, or to obtain references to remote objects.
AppDomain	Represents an application domain, which is equivalent to a thread of execution. Cannot be inherited.

Class*	***Description***
Array	Supports creating, manipulating, searching and sorting arrays. This is the base class for all other array classes in the .NET Framework.
Attribute	Base class for custom attributes.
BitConverter	Converts base data types into an array of bytes and vice-versa. Used often for marshaling/unmarshaling objects between applications or across the network.
Buffer	Represents a chunk of memory as an array of bytes.
CharEnumerator	Provides support for iterating strings character-by-character.
Console	Represents the standard input, output and error streams for console applications. Cannot be inherited.
Convert	Supports converting one base data type into another. Data types include Boolean, Char, SByte, Byte, Int16, Int32, Int64, UInt16, UInt32, UInt64, Single, Double, Decimal, DateTime and String.
Delegate	Base class for all other delegate classes.
Enum	Base class for all other enumerations.
Environment	This class holds environment information including command line strings, current working directory, computer name, and operating system information.
EventArgs	Base class for all event argument classes.
Exception	Base class for all exceptions.
GC	The system garbage collector, which is a service that automatically recovers unused memory in .NET applications. This is a system service and can be controlled using this class.
MarshalByRefObject	Enables accessing objects in other application domains.
Math	Supplies a wide range of mathematical constants and functions.
MulticastDelegate	Base class for delegates that support notification of more than one callback method of an event.
Object	The base class for all other .NET classes.

Class*	***Description***
OperatingSystem	This class represents operating system information including version and platform identifiers.
Random	A pseudo-random number generator.
String	Represents an immutable (i.e. Non-modifiable) series of characters.
TimeZone	Represents time zone information.
Type	Holds information about other types such as classes, data types, enumerations, and array types. Using this class you can query the system to find out details about another data type, including which assembly it is defined by, what attributes, events and methods it supports and much more.
Uri	Represents a URI (Uniform Resource Identifier) and provides methods to access individual parts of the URI such as protocol, host, etc.
ValueType	Base class for value type classes such as Int16, Byte and others.
Version	Represents the version number attribute of an assembly.
WeakReference	Special class that allows you to hold a reference to an object, but still allows the object to be garbage collected if memory is tight.

* This is not a complete list. Consult the documentation for more information.

Interface*	***Description***
ICloneable	Supports cloning, which allows you to create a new instance of an object with the same values as the original.
IComparable	A generalized way for classes to support comparing objects to each other.
IConvertible	Defines methods to convert objects into equivalent CLR types. You can support methods like ToString(), ToInt16(), etc...
IDisposable	Provides methods that can release unmanaged blocks of memory. Similar in concept to destructors.

Interface*	***Description***
IFormattable	Defines methods that support formatted conversion of objects into strings.
IServiceProvider	Defines methods for service objects, which are objects that provide custom support to other objects.

* This is not a complete list. Consult the documentation for more information.

Structure*	***Description***
ArgIterator	Holds a variable list of arguments for methods that can be invoked with different numbers of arguments.
Boolean	A boolean value type.
Byte	An 8-bit unsigned integer.
Char	A Unicode character.
DateTime	An instant in time including both date and time values.
Decimal	A decimal number type usually used for financial calculations. Values range from +79,228,162,514,264,337,593,543,950,335 to -79,228,162,514,264,337,593,543,950,335 and mathematical operations do not have rounding errors.
Double	A double-precision floating-point type.
Guid	A globally unique identifier.
Int16	A 16-bit signed integer.
Int32	A 32-bit signed integer.
Int64	A 64-bit signed integer.
IntPtr	Holds a pointer or handle to platform-specific objects or devices.
SByte	An 8-bit signed integer.
Single	A single-precision floating point type.
TimeSpan	Represents a time interval.
UInt16	A 16-bit unsigned integer.
UInt32	A 32-bit unsigned integer.
UInt64	A 64-bit unsigned integer.
UIntPtr	Holds a pointer or handle to platform-specific objects or devices.

Structure*	***Description***
Void	Used for methods that do not return another data type.

* This is not a complete list. Consult the documentation for more information.

Delegate*	***Description***
AsyncCallback	References a callback method that will be notified when an asynchronous operation completes.
EventHandler	Represents the method that handles an event with no event data.

* This is not a complete list. Consult the documentation for more information.

Enumeration*	***Description***
DayOfWeek	Specifies the day of the week.
Environment.SpecialFolder	Specifies constants used to retrieve paths to system folders.
PlatformID	Describes platforms supported by an assembly.
TypeCode	Specifies the type of an object.

* This is not a complete list. Consult the documentation for more information.

System.CodeDom

Contains classes that can be used to represent the elements and structure of a source code document.

System.CodeDom.Compiler

Contains classes that can be used to manage the generation and compilation of source code in supported programming languages based on the structure of Code Document Object Model (CodeDOM) source code models.

System.Collections

Contains interfaces and classes that define various collections of objects, such as lists, queues, bit arrays, hashtables and dictionaries.

Class	***Description***
ArrayList	Implements the IList interface for arrays that can dynamically grow as needed.

Class	***Description***
BitArray	Manages a compact array of boolean values as a series of bits.
CaseInsensitiveComparer	Compares two objects, ignoring string case.
CaseInsensitiveHashCodeProvider	Supplies hash codes for objects, ignoring string case.
CollectionBase	Abstract base class for the other collection classes.
Comparer	Compares two objects. Case sensitive when used to compare strings.
DictionaryBase	Abstract base class for collections that support named key-value pairs.
Hashtable	A collection of key-value pairs organized based on the hash codes of the keys.
Queue	A first-in, first-out collection.
ReadOnlyCollectionBase	Abstract base class for collections that are read-only.
SortedList	A collection of key-value pairs organized by keys in sorted sequence. Values can be accessed either via the key, or an index.
Stack	A last-in, first-out collection.

Interface	***Description***
ICollection	A common set of methods supported by all collections.
IComparer	Defines methods used to compare two objects.
IDictionary	Methods used to work on key-value pairs.
IDictionaryEnumerator	Methods used to iterate over dictionaries.
IEnumerable	Defines methods that support iteration over collections.
IEnumerator	Supports simple iteration of collections.
IHashCodeProvider	Defines methods used to generated hash codes for objects.

Interface	***Description***
IList	Defines methods to provide index access to collections that work like arrays.

Structure	***Description***
DictionaryEntry	Holds a key-value pair that can be set or retrieved.

System.Collections.Specialized

Contains specialized and strongly typed collections; for example, a linked list dictionary, a bit vector, and collections that contain only strings.

System.ComponentModel

Provides classes that are used to implement the run-time and design-time behavior of components and controls. This namespace includes the base classes and interfaces for implementing attributes and type converters, binding to data sources, and licensing components.

System.ComponentModel.Design

Enables developers to build custom user interface controls and include them in a design-time environment to be used along with vendor controls.

System.ComponentModel.Design.Serialization

Provides support for componentserialization by designers. The classes in this namespace can be used to provide custom serializers, manage the serialization of particular types, manage designer loading and designer serialization, and optimize designer reloading.

System.Configuration

Provides classes and interfaces that allow you to programmatically access .NET Framework configuration settings and handle errors in configuration files (.config files).

Class	***Description***
AppSettingsReader	Supports reading values from the application's .config file.
ConfigurationException	The exception that is thrown if an error occurs in a configuration setting.
ConfigurationSettings	Represents one section of a configuration file.

Class	***Description***
DictionarySectionHandler	Reads key-value pairs from a configuration section.
IgnoreSectionHandler	Provides support for sections in a configuration file not supported by System.Configuration classes.
NameValueSectionHandler	Reads name-value pairs from a configuration section.
SingleTagSectionHandler	Supports reading XML attributes as key-value pairs.

System.Configuration.Assemblies

Contains classes that are used to configure an assembly.

System.Configuration.Install

Provides classes that allow you to write custom installers for your own components. The Installer class is the base class for all custom installers in the .NET Framework.

System.Data

Consists mostly of the classes that constitute the ADO.NET architecture. The ADO.NET architecture enables you to build components that efficiently manage data from multiple data sources. In a disconnected scenario (such as the Internet), ADO.NET provides the tools to request, update, and reconcile data in multiple tier systems. The ADO.NET architecture is also implemented in client applications, such as Windows Forms, or HTML pages created by ASP.NET.

Class*	***Description***
DataColumn	The schema of a column in a DataTable.
DataRelation	Represents a parent/child relationship between two data tables.
DataRow	Represents a row of data from a DataTable.
DataRowView	A special form which is connected to a database table and updates as the current row moves.
DataTable	Represents an in-memory data table.
PropertyCollection	A collection of properties that can be applied to a DataColumn, DataRow, or DataTable.

* This is not a complete list. Consult the documentation for more information.

System.Data.Common

Contains classes shared by the .NET data providers. A .NET data provider describes a collection of classes used to access a data source, such as a database, in the managed space.

System.Data.OleDb

Encapsulates the OLE DB .NET Data Provider. A .NET data provider describes a collection of classes used to access a data source, such as a database, in the managed space.

System.Data.SqlClient

Encapsulates the SQL Server .NET Data Provider. A .NET data provider describes a collection of classes used to access a data source, such as a database, in the managed space.

System.Data.SqlTypes

Provides classes for native data types within SQL Server. These classes provide a safer, faster alternative to other data types. Using the classes in this namespace helps prevent type conversion errors caused in situations where loss of precision could occur.

System.Diagnostics

Provides classes that allow you to interact with system processes, event logs, and performance counters. This namespace also provides classes that allow you to debug your application and to trace the execution of your code.

Classes*	***Description***
Debug	Provides methods to print debugging messages and check for logic errors in the code.
EventLog	Provides support for using the Windows Event Log.
PerformanceCounter	Provides support for accessing and creating Windows Performance Monitor counters.
Process	Provides support for accessing local and remote processes, including the ability to start and stop system services.
ProcessThread	Represents an operating system thread.
Trace	Provides methods and properties that help you trace the execution of code.

* This is not a complete list. Consult the documentation for more information.

System.Diagnostics.SymbolStore

Provides classes that allow you to read and write debug symbol information, such as source line to Microsoft intermediate language (MSIL) maps. Compilers targeting the .NET Framework can store the debug symbol information into a programmer's database (PDB) files. Debuggers and code profiler tools can read the debug symbol information at run time.

System.DirectoryServices

Provides easy access to Active Directory from managed code.

System.Drawing

Provides access to GDI+ basic graphics functionality. More advanced functionality is provided in the System.Drawing.Drawing2D, System.Drawing.Imaging, and System.Drawing.Text namespaces.

Classes*	***Description***
Bitmap	Represents pixel data images.
Brush	A class used for filling and painting areas like rectangles, ellipses, pies and polygons.
Font	Defines a text format including font family, size, and style.
Graphics	Represents a drawing surface. (Like a device context.)
Icon	Represents a Windows icon.
Image	Abstract base class used by the Bitmap, Icon and Metafile classes.
ImageAnimator	Animates images using time-based frames.
Pen	Defines an object used for drawing lines and circles.
Region	Defines the interior of a graphics shape.
StringFormat	Encapsulates text layout information, including alignment, spacing, ellipsis insertion, and national digit substitution.

* This is not a complete list. Consult the documentation for more information.

Structure*	***Description***
Color	An ARGB (alpha, red, green, blue) color.
Point	An integer based x,y coordinate.
PointF	A floating-point based x,y coordinate.

Structure*	***Description***
Rectangle	Stores location and size of rectangles.
Size	Stores height and width of rectangular areas.

* This is not a complete list. Consult the documentation for more information.

System.Drawing.Design

Contains classes that extend design-time user interface (UI) logic and drawing. You can further extend this design-time functionality to create custom toolbox items, type-specific value editors that can edit and graphically represent values of their supported types, or type converters that can convert values between certain types.

System.Drawing.Drawing2D

Provides advanced 2-dimensional and vector graphics functionality. This namespace includes the gradient brushes, the Matrix class (used to define geometric transforms), and the GraphicsPath class.

System.Drawing.Imaging

Provides advanced GDI+ imaging functionality. Basic graphics functionality is provided by the System.Drawing namespace.

System.Drawing.Printing

Provides print-related services.

System.Drawing.Text

Provides advanced GDI+ typography functionality. Basic graphics functionality is provided by the System.Drawing namespace. The classes in this namespace allow users to create and use collections of fonts.

System.EnterpriseServices

Provides an important infrastructure for enterprise applications. COM+ provides a services architecture for component programming models deployed in an enterprise environment. This namespace provides .NET Framework objects with access to COM+ services, making the .NET Framework objects more practical for enterprise applications.

System.EnterpriseServices.CompensatingResourceManager

Provides classes that allow you to use a Compensating Resource Manager (CRM) in managed code. A CRM is a service provided by COM+ that enables you to indude non transactional objects in Microsoft Distributed Transaction Coordinator (DTC) transactions. Although CRMs do not provide the capabilities of a full resource manager, they do provide transactional atomicity (all-or-nothing behavior) and durability through the recovery log.

System.Globalization

Contains classes that define culture-related information, including the language, the country/region, the calendars in use, the format patterns for dates, currency, and numbers, and the sort order for strings.

System.IO

Contains types that allow synchronous and asynchronous reading and writing on data streams and files.

Classes*	***Description***
BinaryReader	Reads primitive data types as binary values.
BinaryWriter	Writes primitive data types as binary values to streams.
BufferedStream	Adds buffering capabilities to other streams.
Directory	Static methods for creating, moving and enumerating through directories and subdirectories.
File	Static methods for creating, copying, moving, deleting and opening files. Works closely with the FileStream class.
FileStream	Allows opening a file as a stream for reading or writing.
FileSystemWatcher	Raises events when files or directories are changed.
MemoryStream	A stream connected to a block of memory instead of a physical file.
Path	Performs operations on a file and/or pathname.
Stream	A generic stream class that represents a series of bytes.
StreamReader	A TextReader used to read characters from a byte stream.
StreamWriter	A TextWriter used to write characters to a byte stream.
StringReader	A TextReader used to read characters from a string.
StringWriter	A TextWriter used to write characters to a string.

Classes*	***Description***
TextReader	A reader that can read characters from a byte stream.
TextWriter	A writer that can write characters to a byte stream.

* This is not a complete list. Consult the documentation for more information.

System.IO.IsolatedStorage

Contains types that allow the creation and use of isolated stores. With these stores, you can read and write data that less trusted code cannot access and prevent the exposure of sensitive information that can be saved elsewhere on the file system. Data is stored in compartments that are isolated by the current user and by the assembly in which the code exists.

System.Management

Provides access to a rich set of management information and management events about the system, devices, and applications instrumented to the Windows Management Instrumentation (WMI) infrastructure.

System.Management.Instrumentation

Provides the classes necessary for instrumenting applications for management and exposing their management information and events through WMI to potential consumers. Consumers such as Microsoft Application Center or Microsoft Operations Manager can then manage your application easily, and monitoring and configuring of your application is available for administrator scripts or other applications, both managed as well as unmanaged.

System.Messaging

Provides classes that allow you to connect to, monitor, and administer message queues on the network and send, receive, or peek messages.

System.Net

Provides a simple programming interface for many of the protocols used on networks today. The WebRequest and WebResponse classes form the basis of what are called pluggable protocols, an implementation of network services that enables you to develop applications that use Internet resources without worrying about the specific details of the individual protocols.

Classes*	***Description***
Authorization	A class used to hold an authentication message for an Internet server.
Cookie	Class used to manage cookies.
Dns	Class used to perform name lookups using the Domain Name System (DNS).
EndPoint	Abstract base class that defines a network address.
HttpWebRequest	HTTP-specific implementation of the generic WebRequest class.
HttpWebResponse	HTTP-specific implementation of the generic WebResponse class.
IPAddress	Represents an IP address.
IPEndPoint	A network endpoint including IP address and port number.
IPHostEntry	Internet host address information.
WebClient	Provides many common methods for sending and receiving data using a URI.
WebRequest	Abstract base class representing a network request.
WebResponse	Abstract base class representing a network response.

* This is not a complete list. Consult the documentation for more information.

System.Net.Sockets

Provides a managed implementation of the Windows Sockets (Winsock) interface for developers who need to tightly control access to the network.

System.Reflection

Contains classes and interfaces that provide a managed view of loaded types, methods, and fields, with the ability to dynamically create and invoke types.

System.Reflection.Emit

Contains classes that allow a compiler or tool to emit metadata and Microsoft intermediate language (MSIL) and optionally generate a PE file on disk. The primary clients of these classes are script engines and compilers.

System.Resources

Provides classes and interfaces that allow developers to create, store, and manage various culture-specific resources used in an application.

System.Runtime.CompilerServices

Provides functionality for compiler writers using managed code to specify attributes in metadata that affect the run-time behavior of the common language runtime. The classes in this namespace are for compiler writers use only.

System.Runtime.InteropServices

Provides a collection of classes useful for accessing COM objects, and native APIs from .NET. The types in this namespace fall into the following areas of functionality: attributes, exceptions, managed definitions of COM types, wrappers, type converters, and the Marshal class.

System.Runtime.InteropServices.Expando

Contains the IExpando interface which allows modification of an object by adding or removing its members.

System.Runtime.Remoting

Provides classes and interfaces that allow developers to create and configure distributed applications.

System.Runtime.Remoting.Activation

Provides classes and objects that support server and client activation of remote objects.

System.Runtime.Remoting.Channels

Contains classes that support and handle channels and channel sinks, which are used as the transport medium when a client calls a method on a remote object.

System.Runtime.Remoting.Channels.Http

Contains channels that use the HTTP protocol to transport messages and objects to and from remote locations. By default, the HTTP channels encode objects and method calls in SOAP format for transmission, but other encoding and decoding formatter sinks can be specified in the configuration properties of a channel.

System.Runtime.Remoting.Channels.Tcp

Contains channels that use the TCP protocol to transport messages and objects to and from remote locations. By default, the TCP channels encode objects and method calls in binary format for transmission, but other encoding and decoding formatter sinks can be specified in the configuration properties of a channel.

System.Runtime.Remoting.Contexts

Contains objects that define the contexts all objects reside within. A context is an ordered sequence of properties that defines an environment for the objects within it. Contexts are created during the activation process for objects that are configured to require certain automatic services such synchronization, transactions, just-in-time (JIT) activation, security, and so on. Multiple objects can live inside a context.

System.Runtime.Remoting.Lifetime

Contains classes that manage the lifetime of remote objects. Traditionally, distributed garbage collection uses reference counts and pinging for control over the lifetime of objects. This works well when there are a few clients per service, but doesn't scale well when there are thousands of clients per service. The remoting lifetime service associates a lease with each service, and deletes a service when its lease time expires. The lifetime service can take on the function of a traditional distributed garbage collector, and it also adjusts well when the numbers of clients per server increases.

System.Runtime.Remoting.Messaging

Contains classes used to create and remote messages. The remoting infrastructure uses messages to communicate with remote objects. Messages are used to transmit remote method calls, to activate remote objects, and to communicate information. A message object carries a set of named properties, including action identifiers, envoy information, and parameters.

System.Runtime.Remoting.Metadata

Contains classes and attributes that can be used to customize generation and processing of SOAP for objects and fields. The classes of this namespace can be used to indicate the SOAPAction, type output, XML element name, and the method XML namespace URI.

System.Runtime.Remoting.Metadata.W3cXsd2001

Contains the XML Schema Definition (XSD) defined by the World Wide Web Consortium (W3C) in 2001. The XML Schema Part2: Data types specification from W3C identifies format and behavior of various data types. This namespace contains wrapper classes for the data types that conform to the W3C specification. All date and time types conform to the ISO standards specification.

System.Runtime.Remoting.MetadataServices

Contains the classes used by the Soapsuds.exe command line tool and the user code to convert metadata to and from XML schema for the remoting infrastructure.

System.Runtime.Remoting.Proxies

Contains classes that control and provide functionality for proxies. A proxy is a local object that is an image of a remote object. Proxies enable clients to access objects across remoting boundaries.

System.Runtime.Remoting.Services

Contains service classes that provide functionality to the .NET Framework.

System.Runtime.Serialization

Contains classes that can be used for serializing and deserializing objects. Serialization is the process of converting an object or a graph of objects into a linear sequence of bytes for either storage or transmission to another location. Deserialization is the process of taking in stored information and recreating objects from it.

System.Runtime.Serialization.Formatters

Provides common enumerations, interfaces, and classes that are used by serialization formatters.

System.Runtime.Serialization.Formatters.Binary

Contains the BinaryFormatter class, which can be used to serialize and deserialize objects in binary format.

System.Runtime.Serialization.Formatters.Soap

Contains the SoapFormatter class, which can be used to serialize and deserialize objects in the SOAP format.

System.Security

Provides the underlying structure of the common language runtime security system, including base classes for permissions.

System.Security.Cryptography

Provides cryptographic services, including secure encoding and decoding of data, as well as many other operations, such as hashing, random number generation, and message authentication.

System.Security.Cryptography.X509Certificates

Contains the common language runtime implementation of the Authenticode X.509 v.3 certificate. This certificate is signed with a private key that uniquely and positively identifies the holder of the certificate.

System.Security.Cryptography.Xml

Contains an XML model for exclusive use within the .NET framework security system. This XML model should not be used for general purposes. This model allows XML objects to be signed with a digital signature.

System.Security.Permissions

Defines classes that control access to operations and resources based on policy.

System.Security.Policy

Contains code groups, membership conditions, and evidence. These three types of classes are used to create the rules applied by the common language runtime security policy system. Evidence classes are the input to security policy and membership conditions are the switches; together these create policy statements and determine the granted permission set. Policy levels and code groups are the structure of the policy hierarchy. Code groups are the encapsulation of a rule and are arranged hierarchically in a policy level.

System.Security.Principal

Defines a principal object that represents the security context under which code is running.

System.ServiceProcess

Provides classes that allow you to implement, install, and control Windows service applications. Services are long-running executables that run without a user interface. Implementing a service involves inheriting from the ServiceBase class and defining specific behavior to process when start, stop, pause, and continue commands are passed in, as well as custom behavior and actions to take when the system shuts down.

System.Text

Contains classes representing ASCII, Unicode, UTF-7, and UTF-8 character encodings; abstract base classes for converting blocks of characters to and from blocks of bytes; and a helper class that manipulates and formats String objects without creating intermediate instances of String.

System.Text.RegularExpressions

Contains classes that provide access to the .NET Framework regular expression engine. The namespace provides regular expression functionality that may be used from any platform or language that runs within the Microsoft .NET Framework.

System.Threading

Provides classes and interfaces that enable multithreaded programming. This namespace includes a ThreadPool class that manages groups of threads, a Timer class that enables a delegate to be called after a specified amount of time, and a Mutex class for synchronizing mutually exclusive threads. This namespace also provides classes for thread scheduling, wait notification, and deadlock resolution.

System.Timers

Provides the Timer component, which allows you to raise an event on a specified interval.

System.Web

Supplies classes and interfaces that enable browser/server communication. This namespace includes the HTTPRequest class that provides extensive information about the current HTTP request, the HTTPResponse class that manages HTTP output to the client, and the HTTPServerUtility object that provides access to server-side utilities and processes. System.Web also includes classes for cookie manipulation, file transfer, exception information, and output cache control.

System.Web.Caching

Provides classes for caching frequently used resources on the server. This includes ASP.NET pages, web services, and user controls. Additionally, a cache dictionary is available for you to store frequently used resources, such as hashtables and other data structures.

System.Web.Configuration

Contains classes that are used to set up ASP.NET configuration.

System.Web.Hosting

Provides the functionality for hosting ASP.NET applications from managed applications outside of Microsoft Internet Information Server (IIS).

System.Web.Mail

Contains classes that enable you to construct and send messages using the CDOSYS Message component. The mail message is delivered through either the SMTP mail service built into Microsoft Windows 2000 or through an arbitrary SMTP server. The classes in this namespace can be used either from ASP.NET or from any managed application.

System.Web.Security

Contains classes that are used to implement ASP.NET security in Web server applications.

System.Web.Services

Consists of the classes that enable you to build and use Web Services. A Web Service is a programmable entity residing on a Web Server exposed via standard Internet protocols.

System.Web.Services.Configuration

Consists of the classes that configure how XML Web services created using ASP.NET run.

System.Web.Services.Description

Consists of the classes that enable you to publicly describe an XML Web service by using the Web Services Description Language (WSDL). Each class in the this namespace corresponds to a specific element in the WSDL specification, and the class hierarchy corresponds to the XML structure of a valid WSDL document.

System.Web.Services.Discovery

Consists of the classes that allows XML Web service clients to locate the available XML Web services on a Web server through a process called XML Web services Discovery.

System.Web.Services.Protocols

Consists of the classes that define the protocols used to transmit data across the wire during the communication between XML Web service clients and XML Web services created using ASP.NET.

System.Web.SessionState

Supplies classes and interfaces that enable storage of data specific to a single client within a Web application on the server. The session state data is used to give the client the appearance of a persistent connection with the application. State information can be stored within local process memory or, for Web farm configurations, out-of-process using either the ASP.NET State Service or a SQL Server database.

System.Web.UI

Provides classes and interfaces that allow you to create controls and pages that will appear in your Web applications as user interface on a Web page. This namespace includes the Control class, which provides all controls, whether HTML, Web, or User controls, with a common set of functionality. It also includes the Page control, which is generated automatically whenever a request is made for a page in your Web application. Also provided are classes which provide the Web Forms Server Controls data binding functionality, the ability to save the view state of a given control or page, as well as parsing functionality for both programmable and literal controls.

System.Web.UI.Design

Contains classes that can be used to extend design-time support for Web Forms.

System.Web.UI.Design.WebControls

Contains classes that can be used to extend design-time support for Web server controls.

System.Web.UI.HtmlControls

Provides classes that allow you to create HTML server controls on a Web page. HTML server controls run on the server and map directly to standard HTML tags supported by all browsers. This allows you to programmatically control the HTML elements on the Web page.

System.Web.UI.WebControls

Contains classes that allow you to create Web server controls on a Web page. Web controls run on the server and include form controls such as buttons and text boxes, as well as special purpose controls such as a calendar. This allows you to programmatically control these elements on a Web page. Web controls are more abstract than HTML controls. Their object model does not necessarily reflect HTML syntax.

System.Windows.Forms

Contains classes for creating Windows-based applications that take full advantage of the rich user interface features available in the Microsoft Windows operating system.

Classes*	***Description***
Application	Provides static methods for managing an application, including starting and stopping an application, processing Windows messages, and getting information about the application.
AxHost	Wrapper class that allows a .NET assembly to use ActiveX controls.
Button	Represents a Windows button control.
CheckBox	Represents a Windows checkbox control.
CheckedListBox	A list with checkboxes.
Clipboard	A class for working with the Windows clipboard for copy, cut, and paste operations.
ComboBox	Represents a Windows combobox.
CommonDialog	Class for working with Windows common dialogs which provide pre-built file open/close, color selection, font selection and other dialog boxes.
Control	Common base class for all other controls.
Cursor	Allows you to control the appearance of the mouse pointer.
DataGrid	A spreadsheet style control connected to a database.
FileDialog	The file open/save common dialog.
FontDialog	The font selection common dialog.
Form	Represents an application window or dialog box.
Label	A static text control.
ListBox	A simple listbox.
ListView	A complex list with four possible view modes.
MainMenu	Represents a menu on a form.
MessageBox	An easy to use pop-up dialog that displays information to the user.
OpenFileDialog	Display an open file dialog.
PictureBox	Displays an image.
PrintDialog	The print common dialog.
ProgressBar	Represents a Windows progress bar.
RadioButton	A radio button control.

Classes*	***Description***
RichTextBox	A text control that supports rich-text with fonts, underlying, bold, italics, etc...
SaveFileDialog	Displays a file save dialog.
ScrollBar	Represents a scroll bar.
Splitter	Provides support for user resizable docked windows.
StatusBar	Represents a status bar attached to a window.
SystemInformation	Provides system information.
TabControl	Manages a set of tab pages.
TextBox	A Windows text box control.
Timer	Provides support for timed events.
ToolBar	Represents a Windows toolbar.
ToolTip	Displays short pop-up help when the mouse hovers over a control.
TreeView	Provides a hierarchal view of labeled items.
UserControl	An empty control used to create other controls.

* This is not a complete list. Consult the documentation for more information.

System.Windows.Forms.Design
Contains classes that can be used to extend design-time support for Windows Forms.
System.Xml
Provides standards-based support for processing XML.
System.Xml.Schema
Provides standards-based support for XML Schemas (XSD).
System.Xml.Serialization
Contains classes that are used to serialize objects into XML format documents or streams.
System.Xml.XPath
Contains the XPath parser and evaluation engine. It supports the W3C XML Path Language (XPath) Version 1.0 Recommendation (www.w3.org/TR/xpath).
System.Xml.Xsl
Provides support for Extensible Stylesheet Transformation (XSLT) transforms. It supports the W3C XSL Transformations (XSLT) Version 1.0 Recommendation (www.w3.org/TR/xslt).