

System.Runtime.InteropServices

Namespace

Reference

Provides a wide variety of members that support COM interop and platform invoke services. If you are unfamiliar with these services, see [Interoperating with Unmanaged Code](#).

Classes

 Expand table

AllowReversePInvokeCallsAttribute	Allows an unmanaged method to call a managed method.
AutomationProxyAttribute	Specifies whether the type should be marshaled using the Automation marshaler or a custom proxy and stub.
BestFitMappingAttribute	Controls whether Unicode characters are converted to the closest matching ANSI characters.
BStrWrapper	Marshals data of type <code>VT_BSTR</code> from managed to unmanaged code. This class cannot be inherited.
ClassInterfaceAttribute	Indicates the type of class interface to be generated for a class exposed to COM, if an interface is generated at all.
CoClassAttribute	Specifies the class identifier of a coclass imported from a type library.
CollectionsMarshal	An unsafe class that provides a set of methods to access the underlying data representations of collections.
ComAliasNameAttribute	Indicates the COM alias for a parameter or field type.
ComAwareEventInfo	Permits late-bound registration of an event handler.
ComCompatibleVersionAttribute	Indicates to a COM client that all classes in the current version of an assembly are compatible with classes in an earlier version of the assembly.
ComConversionLossAttribute	Indicates that information was lost about a class or interface when it was imported from a type library to an assembly.
ComDefaultInterfaceAttribute	Specifies a default interface to expose to COM. This class cannot be inherited.

ComEventInterfaceAttribute	Identifies the source interface and the class that implements the methods of the event interface that is generated when a coclass is imported from a COM type library.
ComEventsHelper	Provides methods that enable .NET delegates that handle events to be added and removed from COM objects.
COMException	The exception that is thrown when an unrecognized HRESULT is returned from a COM method call.
ComImportAttribute	Indicates that the attributed type was previously defined in COM.
ComRegisterFunctionAttribute	Specifies the method to call when you register an assembly for use from COM; this enables the execution of user-written code during the registration process.
ComSourceInterfacesAttribute	Identifies a list of interfaces that are exposed as COM event sources for the attributed class.
ComUnregisterFunctionAttribute	Specifies the method to call when you unregister an assembly for use from COM; this allows for the execution of user-written code during the unregistration process.
ComVisibleAttribute	Controls accessibility of an individual managed type or member, or of all types within an assembly, to COM.
ComWrappers	Class for managing wrappers of COM IUnknown types.
CriticalHandle	Represents a wrapper class for handle resources.
CurrencyWrapper	Wraps objects the marshaler should marshal as a <code>VT_CY</code> .
DefaultCharSetAttribute	Specifies the value of the CharSet enumeration. This class cannot be inherited.
DefaultDllImportSearchPathsAttribute	Specifies the paths that are used to search for DLLs that provide functions for platform invokes.
DefaultParameterValueAttribute	Sets the default value of a parameter when called from a language that supports default parameters. This class cannot be inherited.
DispatchWrapper	Wraps objects the marshaler should marshal as a <code>VT_DISPATCH</code> .
DispIdAttribute	Specifies the COM dispatch identifier (DISPID) of a method, field, or property.
DllImportAttribute	Indicates that the attributed method is exposed by an unmanaged dynamic-link library (DLL) as a static entry point.
DynamicInterfaceCastableImplementationAttribute	Attribute required by any type that is returned by GetInterfaceImplementation(RuntimeTypeHandle) .

ErrorWrapper	Wraps objects the marshaler should marshal as a <code>VT_ERROR</code> .
ExternalException	The base exception type for all COM interop exceptions and structured exception handling (SEH) exceptions.
FieldOffsetAttribute	Indicates the physical position of fields within the unmanaged representation of a class or structure.
GuidAttribute	Supplies an explicit Guid when an automatic GUID is undesirable.
HandleCollector	Tracks outstanding handles and forces a garbage collection when the specified threshold is reached.
ImmutableCollectionsMarshal	An unsafe class that provides a set of methods to access the underlying data representations of immutable collections.
ImportedFromTypeLibAttribute	Indicates that the types defined within an assembly were originally defined in a type library.
InAttribute	Indicates that data should be marshaled from the caller to the callee, but not back to the caller.
InterfaceTypeAttribute	Indicates whether a managed interface is dual, dispatch-only, or <code>IUnknown</code> -only when exposed to COM.
InvalidComObjectException	The exception thrown when an invalid COM object is used.
InvalidOleVariantTypeException	The exception thrown by the marshaler when it encounters an argument of a variant type that can not be marshaled to managed code.
LCIDConversionAttribute	Indicates that a method's unmanaged signature expects a locale identifier (LCID) parameter.
LibraryImportAttribute	Indicates that a source generator should create a function for marshalling arguments instead of relying on the runtime to generate an equivalent marshalling function at run time.
ManagedToNativeComInteropStubAttribute	Provides support for user customization of interop stubs in managed-to-COM interop scenarios.
Marshal	Provides a collection of methods for allocating unmanaged memory, copying unmanaged memory blocks, and converting managed to unmanaged types, as well as other miscellaneous methods used when interacting with unmanaged code.
MarshalAsAttribute	Indicates how to marshal the data between managed and unmanaged code.
MarshalDirectiveException	The exception that is thrown by the marshaler when it encounters a MarshalAsAttribute it does not support.

MemoryMarshal	Provides methods to interoperate with Memory<T> , ReadOnlyMemory<T> , Span<T> , and ReadOnlySpan<T> .
NativeLibrary	Provides APIs for managing native libraries.
NativeMemory	This class contains methods that are mainly used to manage native memory.
OptionalAttribute	Indicates that a parameter is optional.
OutAttribute	Indicates that data should be marshaled from callee back to caller.
PosixSignalContext	Provides data for a PosixSignalRegistration event.
PosixSignalRegistration	Handles a PosixSignal .
PreserveSigAttribute	Indicates that the HRESULT signature transformation that takes place during COM interop calls should be suppressed.
PrimaryInteropAssemblyAttribute	Indicates that the attributed assembly is a primary interop assembly.
ProgIdAttribute	Allows the user to specify the ProgID of a class.
RuntimeEnvironment	Provides a collection of <code>static</code> methods that return information about the common language runtime environment.
RuntimeInformation	Provides information about the .NET runtime installation.
SafeArrayRankMismatchException	The exception thrown when the rank of an incoming <code>SAFEARRAY</code> does not match the rank specified in the managed signature.
SafeArrayTypeMismatchException	The exception thrown when the type of the incoming <code>SAFEARRAY</code> does not match the type specified in the managed signature.
SafeBuffer	Provides a controlled memory buffer that can be used for reading and writing. Attempts to access memory outside the controlled buffer (underruns and overruns) raise exceptions.
SafeHandle	Represents a wrapper class for operating system handles. This class must be inherited.
SEHException	Represents structured exception handling (SEH) errors.
SequenceMarshal	Provides a collection of methods for interoperating with ReadOnlySequence<T> .
StandardOleMarshalObject	Replaces the standard common language runtime (CLR) free-threaded marshaler with the standard OLE STA marshaler.
StructLayoutAttribute	Lets you control the physical layout of the data fields of a class or structure in memory.

SuppressGCTransitionAttribute	Indicates that a garbage collection transition should be skipped when an unmanaged function call is made.
TypeIdentifierAttribute	Provides support for type equivalence.
TypeLibFuncAttribute	Contains the FUNCFLAGS that were originally imported for this method from the COM type library.
TypeLibImportClassAttribute	Specifies which Type exclusively uses an interface. This class cannot be inherited.
TypeLibTypeAttribute	Contains the TYPEFLAGS that were originally imported for this type from the COM type library.
TypeLibVarAttribute	Contains the VARFLAGS that were originally imported for this field from the COM type library.
TypeLibVersionAttribute	Specifies the version number of an exported type library.
UnknownWrapper	Wraps objects the marshaler should marshal as a <code>VT_UNKNOWN</code> .
UnmanagedCallConvAttribute	Specifies the calling convention required to call P/Invoke methods implemented in unmanaged code.
UnmanagedCallersOnlyAttribute	Any method marked with UnmanagedCallersOnlyAttribute can be directly called from native code. The function token can be loaded to a local variable using the address-of operator in C# and passed as a callback to a native method.
UnmanagedFunctionPointerAttribute	Controls the marshaling behavior of a delegate signature passed as an unmanaged function pointer to or from unmanaged code. This class cannot be inherited.
VariantWrapper	Marshals data of type <code>VT_VARIANT</code> <code>VT_BYREF</code> from managed to unmanaged code. This class cannot be inherited.

Structs


 [Expand table](#)

ArrayWithOffset	Encapsulates an array and an offset within the specified array.
CLong	CLong is an immutable value type that represents the <code>long</code> type in C and C++. It is meant to be used as an exchange type at the managed/unmanaged boundary to accurately represent in managed code unmanaged APIs that use the <code>long</code> type. This type has 32-bits of storage on all Windows platforms and 32-bit Unix-based platforms. It has 64-bits of storage on 64-bit Unix platforms.

ComWrappers.ComInterfaceDispatch	An application binary interface for function dispatch of a COM interface.
ComWrappers.ComInterfaceEntry	Interface type and pointer to targeted VTable.
CULong	CULong is an immutable value type that represents the <code>unsigned long</code> type in C and C++. It is meant to be used as an exchange type at the managed/unmanaged boundary to accurately represent in managed code unmanaged APIs that use the <code>unsigned long</code> type. This type has 32-bits of storage on all Windows platforms and 32-bit Unix-based platforms. It has 64-bits of storage on 64-bit Unix platforms.
GCHandle	Provides a way to access a managed object from unmanaged memory.
HandleRef	Wraps a managed object holding a handle to a resource that is passed to unmanaged code using platform invoke.
NFloat	NFloat is an immutable value type that represents a floating type that has the same size as the native integer size. It is meant to be used as an exchange type at the managed/unmanaged boundary to accurately represent in managed code unmanaged APIs that use a type alias for C or C++'s <code>float</code> on 32-bit platforms or <code>double</code> on 64-bit platforms, such as the <code>CGFloat</code> type in libraries provided by Apple.
OSPlatform	Represents an operating system platform.

Interfaces

 Expand table

ICustomAdapter	Provides a way for clients to access the actual object, rather than the adapter object handed out by a custom marshaler.
ICustomFactory	Enables users to write activation code for managed objects that extend MarshalByRefObject .
ICustomMarshaler	Provides custom wrappers for handling method calls.
ICustomQueryInterface	Enables developers to provide a custom, managed implementation of the IUnknown::QueryInterface(REFIID riid, void **ppvObject)  method.
IDynamicInterfaceCastable	Interface used to participate in a type cast failure.

Enums

 Expand table

Architecture	Indicates the processor architecture.
CallingConvention	Specifies the calling convention required to call methods implemented in unmanaged code.
CharSet	Dictates which character set marshaled strings should use.
ClassInterfaceType	Identifies the type of class interface that is generated for a class.
ComInterfaceType	Identifies how to expose an interface to COM.
ComMemberType	Describes the type of a COM member.
CreateComInterfaceFlags	Specifies flags for the GetOrCreateComInterfaceForObject(Object, CreateComInterfaceFlags) method.
CreateObjectFlags	Specifies flags for the GetOrCreateObjectForComInstance(IntPtr, CreateObjectFlags) method.
CustomQueryInterfaceMode	Indicates whether the GetComInterfaceForObject(Object, Type, CustomQueryInterfaceMode) method's IUnknown::QueryInterface calls can use the ICustomQueryInterface interface.
CustomQueryInterfaceResult	Provides return values for the GetInterface(Guid, IntPtr) method.
DllImportSearchPath	Specifies the paths that are used to search for DLLs that provide functions for platform invokes.
GCHandleType	Represents the types of handles the GCHandle type can allocate.
LayoutKind	Controls the layout of an object when exported to unmanaged code.
PosixSignal	Specifies a POSIX signal number.
StringMarshalling	Specifies how strings should be marshalled for generated p/invokes
TypeLibFuncFlags	Describes the original settings of the <code>FUNCFLAGS</code> in the COM type library from where this method was imported.
TypeLibTypeFlags	Describes the original settings of the <code>TYPEFLAGS</code> in the COM type library from which the type was imported.
TypeLibVarFlags	Describes the original settings of the <code>VARFLAGS</code> in the COM type library from which the variable was imported.
UnmanagedType	Identifies how to marshal parameters or fields to unmanaged code.

[VarEnum](#)

Indicates how to marshal the array elements when an array is marshaled from managed to unmanaged code as a [SafeArray](#).

Delegates

[Expand table](#)

[DllImportResolver](#)

Provides a delegate used to resolve native libraries via callback.

Remarks

Members of this namespace provide several categories of functionality, as shown in the following table. Attributes control marshaling behavior, such as how to arrange structures or how to represent strings. The most important attributes are [DllImportAttribute](#), which you use to define platform invoke methods for accessing unmanaged APIs, and [MarshalAsAttribute](#), which you use to specify how data is marshaled between managed and unmanaged memory.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)