System.Runtime.InteropServices.Marshal Class

```
[ILAsm]
.class public abstract sealed Marshal extends System.Object

[C#]
public static class Marshal
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 4.0.0.0
- *Attributes:*
  - CLSCompliantAttribute(true)

Summary

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.

Inherits From: System.Object

Library: RuntimeInfrastructure

Description

The `static` methods defined on the `System.Runtime.InteropServices.Marshal` class are essential to working with unmanaged code. Most methods defined in this class are typically used by developers who want to provide a bridge between the managed and unmanaged programming models. For example, the `System.Runtime.InteropServices.Marshal.StringToHGlobalAnsi` method copies ANSI characters from a specified string (in the managed heap) to a buffer in the unmanaged heap. It also allocates the target heap of the right size.

The `Read` and `Write` methods in the `System.Runtime.InteropServices.Marshal` class support both aligned and unaligned access.

Marshal.SystemDefaultCharSize Field

```
[ILAsm]
.field public static initonly int32 SystemDefaultCharSize

[C#]
public static readonly int SystemDefaultCharSize
```

Summary

Represents the default character size on the system; the default is 2 for Unicode systems and 1 for ANSI or UTF-8 based systems. This field is read-only.

Marshal.SystemMaxDBCSCharSize Field

```
[ILAsm]
.field public static initonly int32 SystemMaxDBCSCharSize

[C#]
public static readonly int SystemMaxDBCSCharSize
```

Summary

Represents the maximum size of a double byte character set (DBCS) size, in bytes, for the current operating system. This field is read-only.

Marshal.AllocHGlobal(System.Int32) Method

```
[ILAsm]
.method public hidebysig static native int AllocHGlobal(int32 cb) cil
managed

[C#]
public static IntPtr AllocHGlobal (int cb)
```

Summary

Allocates the specified number of bytes from the unmanaged memory of the process.

Parameters

Parameter	Description
cb	The required number of bytes in memory.

Return Value

A pointer to the newly allocated memory. This memory must be released using the `System.Runtime.InteropServices.Marshal.FreeHGlobal` method.

Description

`System.Runtime.InteropServices.Marshal.AllocHGlobal` is the memory allocation method in the `System.Runtime.InteropServices.Marshal` class.

The allocated memory is not zero-filled.

Exceptions

Exception	Condition
System.OutOfMemoryException	There is insufficient memory to satisfy the request.

Marshal.AllocHGlobal(System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static native int AllocHGlobal(native int cb) cil
managed

[C#]
public static IntPtr AllocHGlobal (IntPtr cb)
```

Summary

Allocates the specified number of bytes from the unmanaged memory of the process.

Parameters

Parameter	Description
cb	The required number of bytes in memory.

Return Value

A pointer to the newly allocated memory. This memory must be released using the `System.Runtime.InteropServices.Marshal.FreeHGlobal` method.

Description

`System.Runtime.InteropServices.Marshal.AllocHGlobal` is the allocation method in the `System.Runtime.InteropServices.Marshal` class.

The allocated memory is not zero-filled.

Exceptions

Exception	Condition
System.OutOfMemoryException	There is insufficient memory to satisfy the request.

Marshal.Copy(System.Byte[], System.Int32, System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(uint8[] source, int32
startIndex, native int destination, int32 length) cil managed

[C#]
public static void Copy (byte[] source, int startIndex, IntPtr
destination, int length)
```

Summary

Copies data from a one-dimensional, managed 8-bit unsigned integer array to an unmanaged memory pointer.

Parameters

Parameter	Description
source	The one-dimensional array to copy from.
startIndex	The zero-based index in the source array where copying should start.
destination	The memory pointer to copy to.
length	The number of array elements to copy.

Description

You can use this method to copy a subset of a one-dimensional managed array to an unmanaged C-style array.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* and *length* are not valid.
System.ArgumentNullException	*source*, *startIndex*, *destination*, or *length* is `null`.

Marshal.Copy(System.Char[], System.Int32, System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(char[] source, int32 startIndex,
native int destination, int32 length) cil managed

[C#]
public static void Copy (char[] source, int startIndex, IntPtr
destination, int length)
```

Summary

Copies data from a one-dimensional, managed character array to an unmanaged memory pointer.

Parameters

Parameter	Description
source	The one-dimensional array to copy from.
startIndex	The zero-based index in the source array where copying should start.
destination	The memory pointer to copy to.
length	The number of array elements to copy.

Description

You can use this method to copy a subset of a one-dimensional managed array to an unmanaged C-style array.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* and *length* are not valid.
System.ArgumentNullException	*startIndex*, *destination*, or *length* is `null`.

Marshal.Copy(System.Double[], System.Int32, System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(float64[] source, int32
startIndex, native int destination, int32 length) cil managed

[C#]
public static void Copy (double[] source, int startIndex, IntPtr
destination, int length)
```

Summary

Copies data from a one-dimensional, managed double-precision floating-point number array to an unmanaged memory pointer.

Parameters

Parameter	Description
source	The one-dimensional array to copy from.
startIndex	The zero-based index in the source array where copying should start.
destination	The memory pointer to copy to.
length	The number of array elements to copy.

Description

You can use this method to copy a subset of a one-dimensional managed array to an unmanaged C-style array.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* and *length* are not valid.
System.ArgumentNullException	*source*, *startIndex*, *destination*, or *length* is `null`.

1

Marshal.Copy(System.Int16[], System.Int32, System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(int16[] source, int32
startIndex, native int destination, int32 length) cil managed

[C#]
public static void Copy (short[] source, int startIndex, IntPtr
destination, int length)
```

Summary

Copies data from a one-dimensional, managed 16-bit signed integer array to an unmanaged memory pointer.

Parameters

Parameter	Description
source	The one-dimensional array to copy from.
startIndex	The zero-based index in the source array where copying should start.
destination	The memory pointer to copy to.
length	The number of array elements to copy.

Description

You can use this method to copy a subset of a one-dimensional managed array to an unmanaged C-style array.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* and *length* are not valid.
System.ArgumentNullException	*source*, *startIndex*, *destination*, or *length* is `null`.

Marshal.Copy(System.Int32[], System.Int32, System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(int32[] source, int32
startIndex, native int destination, int32 length) cil managed

[C#]
public static void Copy (int[] source, int startIndex, IntPtr destination,
int length)
```

Summary

Copies data from a one-dimensional, managed 32-bit signed integer array to an unmanaged memory pointer.

Parameters

Parameter	Description
source	The one-dimensional array to copy from.
startIndex	The zero-based index in the source array where copying should start.
destination	The memory pointer to copy to.
length	The number of array elements to copy.

Description

You can use this method to copy a subset of a one-dimensional managed array to an unmanaged C-style array.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* and *length* are not valid.
System.ArgumentNullException	*startIndex* or *length* is `null`.

Marshal.Copy(System.Int64[], System.Int32, System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(int64[] source, int32
startIndex, native int destination, int32 length) cil managed

[C#]
public static void Copy (long[] source, int startIndex, IntPtr
destination, int length)
```

Summary

Copies data from a one-dimensional, managed 64-bit signed integer array to an unmanaged memory pointer.

Parameters

Parameter	Description
source	The one-dimensional array to copy from.
startIndex	The zero-based index in the source array where copying should start.
destination	The memory pointer to copy to.
length	The number of array elements to copy.

Description

You can use this method to copy a subset of a one-dimensional managed array to an unmanaged C-style array.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* and *length* are not valid.
System.ArgumentNullException	*source*, *startIndex*, *destination*, or *length* is `null`.

Marshal.Copy(System.IntPtr, System.Byte[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(native int source, uint8[]
destination, int32 startIndex, int32 length) cil managed

[C#]
public static void Copy (IntPtr source, byte[] destination, int
startIndex, int length)
```

Summary

Copies data from an unmanaged memory pointer to a managed 8-bit unsigned integer array.

Parameters

Parameter	Description
source	The memory pointer to copy from.
destination	The array to copy to.
startIndex	The zero-based index in the source array where copying should start.
length	The number of array elements to copy.

Description

Unmanaged, C-style arrays do not contain bounds information, which prevents the *startIndex* and *length* parameters from being validated. Thus, the unmanaged data corresponding to the *source* parameter populates the managed array regardless of its usefulness. You must initialize the managed array with the appropriate size before calling this method.

Exceptions

Exception	Condition
System.ArgumentNullException	*source*, *destination*, *startIndex*, or *length* is `null`.

Marshal.Copy(System.IntPtr, System.Char[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(native int source, char[]
destination, int32 startIndex, int32 length) cil managed

[C#]
public static void Copy (IntPtr source, char[] destination, int
startIndex, int length)
```

Summary

Copies data from an unmanaged memory pointer to a managed character array.

Parameters

Parameter	Description
source	The memory pointer to copy from.
destination	The array to copy to.
startIndex	The zero-based index in the source array where copying should start.
length	The number of array elements to copy.

Description

Unmanaged, C-style arrays do not contain bounds information, which prevents the *startIndex* and *length* parameters from being validated. Thus, the unmanaged data corresponding to the *source* parameter populates the managed array regardless of its usefulness. You must initialize the managed array with the appropriate size before calling this method.

Exceptions

Exception	Condition
System.ArgumentNullException	*source*, *destination*, *startIndex*, or *length* is `null`.

Marshal.Copy(System.IntPtr, System.Double[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(native int source, float64[]
destination, int32 startIndex, int32 length) cil managed

[C#]
public static void Copy (IntPtr source, double[] destination, int
startIndex, int length)
```

Summary

Copies data from an unmanaged memory pointer to a managed double-precision floating-point number array.

Parameters

Parameter	Description
source	The memory pointer to copy from.
destination	The array to copy to.
startIndex	The zero-based index in the source array where copying should start.
length	The number of array elements to copy.

Description

Unmanaged, C-style arrays do not contain bounds information, which prevents the *startIndex* and *length* parameters from being validated. Thus, the unmanaged data corresponding to the *source* parameter populates the managed array regardless of its usefulness. You must initialize the managed array with the appropriate size before calling this method.

Exceptions

Exception	Condition
System.ArgumentNullException	*source*, *destination*, *startIndex*, or *length* is `null`.

Marshal.Copy(System.IntPtr, System.Int16[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(native int source, int16[]
destination, int32 startIndex, int32 length) cil managed

[C#]
public static void Copy (IntPtr source, short[] destination, int
startIndex, int length)
```

Summary

Copies data from an unmanaged memory pointer to a managed 16-bit signed integer array.

Parameters

Parameter	Description
source	The memory pointer to copy from.
destination	The array to copy to.
startIndex	The zero-based index in the source array where copying should start.
length	The number of array elements to copy.

Description

Unmanaged, C-style arrays do not contain bounds information, which prevents the *startIndex* and *length* parameters from being validated. Thus, the unmanaged data corresponding to the *source* parameter populates the managed array regardless of its usefulness. You must initialize the managed array with the appropriate size before calling this method.

Exceptions

Exception	Condition
System.ArgumentNullException	*source*, *destination*, *startIndex*, or *length* is `null`.

Marshal.Copy(System.IntPtr, System.Int32[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(native int source, int32[]
destination, int32 startIndex, int32 length) cil managed

[C#]
public static void Copy (IntPtr source, int[] destination, int startIndex,
int length)
```

Summary

Copies data from an unmanaged memory pointer to a managed 32-bit signed integer array.

Parameters

Parameter	Description
source	The memory pointer to copy from.
destination	The array to copy to.
startIndex	The zero-based index in the source array where copying should start.
length	The number of array elements to copy.

Description

Unmanaged, C-style arrays do not contain bounds information, which prevents the *startIndex* and *length* parameters from being validated. Thus, the unmanaged data corresponding to the *source* parameter populates the managed array regardless of its usefulness. You must initialize the managed array with the appropriate size before calling this method.

Exceptions

Exception	Condition
System.ArgumentNullException	*source*, *destination*, *startIndex*, or *length* is `null`.

Marshal.Copy(System.IntPtr, System.Int64[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(native int source, int64[]
destination, int32 startIndex, int32 length) cil managed

[C#]
public static void Copy (IntPtr source, long[] destination, int
startIndex, int length)
```

Summary

Copies data from an unmanaged memory pointer to a managed 64-bit signed integer array.

Parameters

Parameter	Description
source	The memory pointer to copy from.
destination	The array to copy to.
startIndex	The zero-based index in the source array where copying should start.
length	The number of array elements to copy.

Description

Unmanaged, C-style arrays do not contain bounds information, which prevents the *startIndex* and *length* parameters from being validated. Thus, the unmanaged data corresponding to the *source* parameter populates the managed array regardless of its usefulness. You must initialize the managed array with the appropriate size before calling this method.

Exceptions

Exception	Condition
System.ArgumentNullException	*source*, *destination*, *startIndex*, or *length* is `null`.

Marshal.Copy(System.IntPtr, System.IntPtr[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(native int source, native int[]
destination, int32 startIndex, int32 length) cil managed

[C#]
public static void Copy (IntPtr source, IntPtr[] destination, int
startIndex, int length)
```

Summary

Copies data from an unmanaged memory pointer to a managed `System.IntPtr` array.

Parameters

Parameter	Description
source	The memory pointer to copy from.
destination	The array to copy to.
startIndex	The zero-based index into the array where copying should start.
length	The number of array elements to copy.

Description

Unmanaged, C-style arrays do not contain bounds information, which prevents the *startIndex* and *length* parameters from being validated. Therefore, the unmanaged data that corresponds to the *source* parameter populates the managed array regardless of its usefulness. You must initialize the managed array with the appropriate size before calling the `System.Runtime.InteropServices.Marshal.Copy` method.

Exceptions

Exception	Condition
System.ArgumentNullException	*source*, *destination*, *startIndex*, or *length* is `null`.

Marshal.Copy(System.IntPtr, System.Single[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(native int source, float32[]
destination, int32 startIndex, int32 length) cil managed

[C#]
public static void Copy (IntPtr source, float[] destination, int
startIndex, int length)
```

Summary

Copies data from an unmanaged memory pointer to a managed single-precision floating-point number array.

Parameters

Parameter	Description
source	The memory pointer to copy from.
destination	The array to copy to.
startIndex	The zero-based index in the source array where copying should start.
length	The number of array elements to copy.

Description

Unmanaged, C-style arrays do not contain bounds information, which prevents the *startIndex* and *length* parameters from being validated. Thus, the unmanaged data corresponding to the *source* parameter populates the managed array regardless of its usefulness. You must initialize the managed array with the appropriate size before calling this method.

Exceptions

Exception	Condition
System.ArgumentNullException	*source*, *destination*, *startIndex*, or *length* is `null`.

Marshal.Copy(System.IntPtr[], System.Int32, System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(native int[] source, int32
startIndex, native int destination, int32 length) cil managed

[C#]
public static void Copy (IntPtr[] source, int startIndex, IntPtr
destination, int length)
```

Summary

Copies data from a one-dimensional, managed `System.IntPtr` array to an unmanaged memory pointer.

Parameters

Parameter	Description
source	The one-dimensional array to copy from.
startIndex	The zero-based index into the array where copying should start.
destination	The memory pointer to copy to.
length	The number of array elements to copy.

Description

You can use this method to copy a subset of a one-dimensional managed `System.IntPtr` array to an unmanaged C-style array.

Exceptions

Exception	Condition
System.ArgumentNullException	*source*, *destination*, *startIndex*, or *length* is `null`.

Marshal.Copy(System.Single[], System.Int32, System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void Copy(float32[] source, int32
startIndex, native int destination, int32 length) cil managed

[C#]
public static void Copy (float[] source, int startIndex, IntPtr
destination, int length)
```

Summary

Copies data from a one-dimensional, managed single-precision floating-point number array to an unmanaged memory pointer.

Parameters

Parameter	Description
source	The one-dimensional array to copy from.
startIndex	The zero-based index in the source array where copying should start.
destination	The memory pointer to copy to.
length	The number of array elements to copy.

Description

You can use this method to copy a subset of a one-dimensional managed array to an unmanaged C-style array.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* and *length* are not valid.
System.ArgumentNullException	*source*, *startIndex*, *destination*, or *length* is `null`.

1

Marshal.DestroyStructure(System.IntPtr, System.Type) Method

```
[ILAsm]
.method public hidebysig static void DestroyStructure(native int ptr,
class System.Type structuretype) cil managed internalcall

[C#]
public static void DestroyStructure (IntPtr ptr, Type structuretype)
```

Summary

Frees all substructures that the specified unmanaged memory block points to.

Parameters

Parameter	Description
ptr	A pointer to an unmanaged block of memory.
structuretype	Type of a formatted class. This provides the layout information necessary to delete the buffer in the *ptr* parameter.

Description

You can use this method to free reference-type fields, such as strings, of an unmanaged structure. Unlike its fields, a structure can be a value type or a reference type. Value-type structures that contain value-type fields (all blittable) have no references whose memory must be freed. The `System.Runtime.InteropServices.Marshal.StructureToPtr` method uses this method to prevent memory leaks when reusing memory occupied by a structure.

`System.Runtime.InteropServices.Marshal.DestroyStructure` calls a platform-specific function, which, in turn, frees an allocated string.

In addition to `System.Runtime.InteropServices.Marshal.DestroyStructure`, the `System.Runtime.InteropServices.Marshal` class provides the `System.Runtime.InteropServices.Marshal.FreeHGlobal` memory deallocation method.

Exceptions

Exception	Condition
System.ArgumentException	*structureType* has an automatic layout. Use sequential or

	explicit instead.

1

2

Marshal.FreeHGlobal(System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static void FreeHGlobal(native int hglobal) cil
managed

[C#]
public static void FreeHGlobal (IntPtr hglobal)
```

Summary

Frees memory previously allocated from the unmanaged memory of the process.

Parameters

Parameter	Description
hglobal	The handle returned by the original matching call to `System.Runtime.InteropServices.Marshal. AllocHGlobal`.

Description

You can use `System.Runtime.InteropServices.Marshal.FreeHGlobal` to free any memory from the global heap allocated by `System.Runtime.InteropServices.Marshal.AllocHGlobal` or `System.Runtime.InteropServices.Marshal.ReAllocHGlobal`. If the *hglobal* parameter is `System.IntPtr.Zero` the method does nothing.

In addition to `System.Runtime.InteropServices.Marshal.FreeHGlobal`, the `System.Runtime.InteropServices.Marshal` class provides another memory-deallocation method: `System.Runtime.InteropServices.Marshal.DestroyStructure`.

Marshal.GetDelegateForFunctionPointer(System.IntPtr, System.Type) Method

```
[ILAsm]
.method public hidebysig static class System.Delegate
GetDelegateForFunctionPointer(native int ptr, class System.Type t) cil
managed

[C#]
public static Delegate GetDelegateForFunctionPointer (IntPtr ptr, Type t)
```

Summary

Converts an unmanaged function pointer to a delegate.

Parameters

Parameter	Description
ptr	The unmanaged function pointer to be converted.
t	The type of the delegate to be returned.

Return Value

A delegate instance that can be cast to the appropriate delegate type.

Description

One can use the `System.Runtime.InteropServices.Marshal.GetDelegateForFunctionPointer` and `System.Runtime.InteropServices.Marshal.GetFunctionPointerForDelegate` methods to marshal delegates in both directions. With `System.Runtime.InteropServices.Marshal.GetDelegateForFunctionPointer`, *ptr* is imported as a `System.IntPtr`. A `System.IntPtr` can be obtained for a managed delegate by calling `System.Runtime.InteropServices.Marshal.GetFunctionPointerForDelegate` and passed as a parameter; it can then be called from inside the unmanaged method. Note that the parameter marshaler can also marshal function pointers to delegates.

The `System.Runtime.InteropServices.Marshal.GetDelegateForFunctionPointer` method has the following restrictions:

- Generics are not supported in interop scenarios.
- You cannot pass an invalid function pointer to this method.

- You can use this method only for pure unmanaged function pointers.
- You cannot use this method with function pointers obtained through C++ or from the `System.RuntimeMethodHandle.GetFunctionPointer` method.
- You cannot use this method to create a delegate from a function pointer to another managed delegate.

Exceptions

Exception	Condition
System.ArgumentException	The *t* parameter is not a delegate or is generic.
System.ArgumentNullException	The *ptr* parameter is `null`. -or- The *t* parameter is `null`.

Marshal.GetFunctionPointerForDelegate(System.Delegate) Method

```
[ILAsm]
.method public hidebysig static native int
GetFunctionPointerForDelegate(class System.Delegate d) cil managed

[C#]
public static IntPtr GetFunctionPointerForDelegate (Delegate d)
```

Summary

Converts a delegate into a function pointer that is callable from unmanaged code.

Parameters

Parameter	Description
d	The delegate to be passed to unmanaged code.

Return Value

A value that can be passed to unmanaged code, which, in turn, can use it to call the underlying managed delegate.

Description

The delegate *d* is converted to a function pointer that can be passed to unmanaged code using the __stdcall calling convention.

You must manually keep the delegate from being collected by the garbage collector from managed code. The garbage collector does not track reference to unmanaged code.

[*Note:* Generics are not supported in interop scenarios.

]

Exceptions

Exception	Condition
System.ArgumentException	The *d* parameter is a generic type.
System.ArgumentNullException	The *d* parameter is `null`.

1

2

Marshal.GetLastWin32Error() Method

```
[ILAsm]
.method public hidebysig static int32 GetLastWin32Error() cil managed
internalcall

[C#]
public static int GetLastWin32Error ()
```

Summary

Returns the error code returned by the last unmanaged function that was called using platform invoke that has the `System.Runtime.InteropServices.DllImportAttribute.SetLastError` flag set.

Return Value

The last error code set by a call to the platform-specific error function.

Description

`System.Runtime.InteropServices.Marshal.GetLastWin32Error` exposes platform-specific error codes. This method exists because it is not safe to make a direct platform invoke call to `GetLastError` to obtain this information. If you want to access this error code, you must call `System.Runtime.InteropServices.Marshal.GetLastWin32Error` instead of writing your own platform invoke definition for `GetLastError` and calling it.

You can use this method to obtain error codes only if you apply the `System.Runtime.InteropServices.DllImportAttribute` to the method signature and set the `System.Runtime.InteropServices.DllImportAttribute.SetLastError` field to `true`.

[*Note:* The name of this method reflects the platform-specificity of the original implementation, and is preserved for compatibility. On other platforms this method may be used to get the last platform-specific error.

]

Marshal.OffsetOf(System.Type, System.String) Method

```
[ILAsm]
.method public hidebysig static native int OffsetOf(class System.Type t,
string fieldName) cil managed

[C#]
public static IntPtr OffsetOf (Type t, string fieldName)
```

Summary

Returns the field offset of the unmanaged form of the managed class.

Parameters

Parameter	Description
t	A value type or formatted reference type that specifies the managed class. You must apply the `System.Runtime.InteropServices. StructLayoutAttribute` to the class.
fieldName	The field within the *t* parameter.

Return Value

The offset, in bytes, for the *fieldName* parameter within the specified class that is declared by platform invoke.

Description

`System.Runtime.InteropServices.Marshal.OffsetOf` provides the offset in terms of the unmanaged structure layout, which does not necessarily correspond to the offset of the managed structure layout. Marshaling the structure can transform the layout and alter the offset. The *t* parameter can be a value type or a formatted reference type (with either a sequential or explicit layout). You can obtain the size of the entire layout by using the `System.Runtime.InteropServices.Marshal.SizeOf` method.

Exceptions

Exception	Condition
System.ArgumentException	The class cannot be exported as a structure.
System.ArgumentNullException	The *t* parameter is `null`.

1

2

Marshal.PtrToStringAnsi(System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static string PtrToStringAnsi(native int ptr) cil
managed

[C#]
public static string PtrToStringAnsi (IntPtr ptr)
```

Summary

Copies all characters up to the first null character from an unmanaged ANSI string to a managed `System.String`, and widens each ANSI character to Unicode.

Parameters

Parameter	Description
ptr	The address of the first character of the unmanaged string.

Return Value

A managed string that holds a copy of the unmanaged ANSI string. If *ptr* is `null`, the method returns a null string.

Description

`System.Runtime.InteropServices.Marshal.PtrToStringAnsi` is useful for custom marshaling or when mixing managed and unmanaged code. Because this method creates a copy of the unmanaged string's contents, you must free the original string as appropriate. This method provides the opposite functionality of the `System.Runtime.InteropServices.Marshal.StringToHGlobalAnsi` method.

Marshal.PtrToStringAnsi(System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static string PtrToStringAnsi(native int ptr,
int32 len) cil managed internalcall

[C#]
public static string PtrToStringAnsi (IntPtr ptr, int len)
```

Summary

Allocates a managed `System.String`, copies a specified number of characters from an unmanaged ANSI string into it, and widens each ANSI character to Unicode.

Parameters

Parameter	Description
ptr	The address of the first character of the unmanaged string.
len	The byte count of the input string to copy.

Return Value

A managed string that holds a copy of the native ANSI string if the value of the *ptr* parameter is not `null`; otherwise, this method returns `null`.

Description

`System.Runtime.InteropServices.Marshal.PtrToStringAnsi` is useful for custom marshaling or when mixing managed and unmanaged code. Because this method creates a copy of the unmanaged string's contents, you must free the original string as appropriate. This method provides the opposite functionality of the `System.Runtime.InteropServices.Marshal.StringToHGlobalAnsi` method.

Exceptions

Exception	Condition
System.ArgumentException	*len* is less than zero.

Marshal.PtrToStringAuto(System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static string PtrToStringAuto(native int ptr) cil
managed

[C#]
public static string PtrToStringAuto (IntPtr ptr)
```

Summary

Allocates a managed `System.String` and copies all characters up to the first null character from a string stored in unmanaged memory into it.

Parameters

Parameter	Description
ptr	For Unicode platforms, the address of the first Unicode character.-or- For ANSI plaforms, the address of the first ANSI character.

Return Value

A managed string that holds a copy of the unmanaged string if the value of the *ptr* parameter is not `null`; otherwise, this method returns `null`.

Description

If the current platform is Unicode, each ANSI character is widened to a Unicode character and this method calls `System.Runtime.InteropServices.Marshal.PtrToStringUni`. Otherwise, this method calls `System.Runtime.InteropServices.Marshal.PtrToStringAnsi`.

`System.Runtime.InteropServices.Marshal.PtrToStringAuto` is useful for custom marshaling or when mixing managed and unmanaged code. Because this method creates a copy of the unmanaged string's contents, you must free the original string as appropriate. `System.Runtime.InteropServices.Marshal.PtrToStringAuto` provides the opposite functionality of the `System.Runtime.InteropServices.Marshal.StringToHGlobalAuto` method.

Marshal.PtrToStringAuto(System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static string PtrToStringAuto(native int ptr,
int32 len) cil managed

[C#]
public static string PtrToStringAuto (IntPtr ptr, int len)
```

Summary

Allocates a managed `System.String` and copies the specified number of characters from a string stored in unmanaged memory into it.

Parameters

Parameter	Description
ptr	For Unicode platforms, the address of the first Unicode character.-or- For ANSI plaforms, the address of the first ANSI character.
len	The number of characters to copy.

Return Value

A managed string that holds a copy of the native string if the value of the *ptr* parameter is not `null`; otherwise, this method returns `null`.

Description

On Unicode platforms, this method calls `System.Runtime.InteropServices.Marshal.PtrToStringUni`; on ANSI platforms, it calls `System.Runtime.InteropServices.Marshal.PtrToStringAnsi`. No transformations are done before these methods are called.

`System.Runtime.InteropServices.Marshal.PtrToStringAuto` is useful for custom marshaling or when mixing managed and unmanaged code. Because this method creates a copy of the unmanaged string's contents, you must free the original string as appropriate. `System.Runtime.InteropServices.Marshal.PtrToStringAuto` provides the opposite functionality of `System.Runtime.InteropServices.Marshal.StringToHGlobalAuto`.

Exceptions

Exception	Condition

System.ArgumentException	*len* is less than zero.

1

2

Marshal.PtrToStringUni(System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static string PtrToStringUni(native int ptr) cil
managed

[C#]
public static string PtrToStringUni (IntPtr ptr)
```

Summary

Allocates a managed `System.String` and copies all characters up to the first null character from an unmanaged Unicode string into it.

Parameters

Parameter	Description
ptr	The address of the first character of the unmanaged string.

Return Value

A managed string that holds a copy of the unmanaged string if the value of the *ptr* parameter is not `null`; otherwise, this method returns `null`.

Description

`System.Runtime.InteropServices.Marshal.PtrToStringUni` is useful for custom marshaling or for use when mixing managed and unmanaged code. Because this method creates a copy of the unmanaged string's contents, you must free the original string as appropriate. This method provides the opposite functionality of the `System.Runtime.InteropServices.Marshal.StringToHGlobalUni` method.

Marshal.PtrToStringUni(System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static string PtrToStringUni(native int ptr,
int32 len) cil managed internalcall

[C#]
public static string PtrToStringUni (IntPtr ptr, int len)
```

Summary

Allocates a managed `System.String` and copies a specified number of characters from an unmanaged Unicode string into it.

Parameters

Parameter	Description
ptr	The address of the first character of the unmanaged string.
len	The number of Unicode characters to copy.

Return Value

A managed string that holds a copy of the unmanaged string if the value of the *ptr* parameter is not `null`; otherwise, this method returns `null`.

Description

`System.Runtime.InteropServices.Marshal.PtrToStringUni` is useful for custom marshaling or when mixing managed and unmanaged code. Because this method creates a copy of the unmanaged string's contents, you must free the original string as appropriate. This method provides the opposite functionality of the `System.Runtime.InteropServices.Marshal.StringToHGlobalUni` method.

Marshal.PtrToStructure(System.IntPtr, System.Object) Method

```
[ILAsm]
.method public hidebysig static void PtrToStructure(native int ptr, object
structure) cil managed

[C#]
public static void PtrToStructure (IntPtr ptr, object structure)
```

Summary

Marshals data from an unmanaged block of memory to a managed object.

Parameters

Parameter	Description
ptr	A pointer to an unmanaged block of memory.
structure	The object to which the data is to be copied. This must be an instance of a formatted class.

Description

`System.Runtime.InteropServices.Marshal.PtrToStructure` is often necessary in interop scenarios when structure parameters are represented as an `System.IntPtr` value. You cannot use this overload method with value types.

Exceptions

Exception	Condition
System.ArgumentException	Structure layout is not sequential or explicit. -or- Structure is a boxed value type.

Marshal.PtrToStructure(System.IntPtr, System.Type) Method

```
[ILAsm]
.method public hidebysig static object PtrToStructure(native int ptr,
class System.Type structureType) cil managed

[C#]
public static object PtrToStructure (IntPtr ptr, Type structureType)
```

Summary

Marshals data from an unmanaged block of memory to a newly allocated managed object of the specified type.

Parameters

Parameter	Description
ptr	A pointer to an unmanaged block of memory.
structureType	The type of object to be created. This object must represent a formatted class or a structure.

Return Value

A managed object containing the data pointed to by the *ptr* parameter.

Description

`System.Runtime.InteropServices.Marshal.PtrToStructure` is often necessary in interop scenarios invoke when structure parameters are represented as an `System.IntPtr` value. You can pass a value type to this overload method. In this case, the returned object is a boxed instance.

Exceptions

Exception	Condition
System.ArgumentException	The *structureType* parameter layout is not sequential or explicit. -or- The *structureType* parameter is a generic type.

System.ArgumentNullException	*structureType* is `null`.

1

2

Marshal.ReadByte(System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static uint8 ReadByte(native int ptr) cil managed

[C#]
public static byte ReadByte (IntPtr ptr)
```

Summary

Reads a single byte from unmanaged memory.

Parameters

Parameter	Description
ptr	The address in unmanaged memory from which to read.

Return Value

The byte read from unmanaged memory.

Description

`System.Runtime.InteropServices.Marshal.ReadByte` has an implied offset of 0. This method enables direct interaction with an unmanaged C-style byte array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	*ptr* is not a recognized format. -or- *ptr* is `null`. -or- *ptr* is invalid.

1

2

Marshal.ReadByte(System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static uint8 ReadByte(native int ptr, int32 ofs)
cil managed

[C#]
public static byte ReadByte (IntPtr ptr, int ofs)
```

Summary

Reads a single byte at a given offset (or index) from unmanaged memory.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory from which to read.
ofs	An additional byte offset, which is added to the *ptr* parameter before reading.

Return Value

The byte read from unmanaged memory at the given offset.

Description

`System.Runtime.InteropServices.Marshal.ReadByte` enables direct interaction with an unmanaged C-style byte array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.

Marshal.ReadByte(System.Object, System.Int32) Method

```
[ILAsm]
.method public hidebysig static pinvokeimpl("mscoree.dll" as "ND_RU1"
winapi) uint8 ReadByte([in] object marshal(as any) ptr, int32 ofs) cil
managed

[C#]
public static byte ReadByte (object ptr, int ofs)
```

Summary

Reads a single byte at a given offset (or index) from unmanaged memory.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory of the source object.
ofs	An additional byte offset, which is added to the *ptr* parameter before reading.

Return Value

The byte read from unmanaged memory at the given offset.

Description

`System.Runtime.InteropServices.Marshal.ReadByte` enables direct interaction with an unmanaged C-style byte array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.
System.ArgumentException	*ptr* is an `System.Runtime.InteropServices.ArrayWithOffset` object. This method does not accept

	`System.Runtime.InteropServices.ArrayWithOffset` parameters.

1

2

Marshal.ReadInt16(System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static int16 ReadInt16(native int ptr) cil
managed

[C#]
public static short ReadInt16 (IntPtr ptr)
```

Summary

Reads a 16-bit signed integer from unmanaged memory.

Parameters

Parameter	Description
ptr	The address in unmanaged memory from which to read.

Return Value

The 16-bit signed integer read from unmanaged memory.

Description

`System.Runtime.InteropServices.Marshal.ReadInt16` has an implied offset of 0. This method enables direct interaction with an unmanaged C-style `Int16` array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	*ptr* is not a recognized format. -or- *ptr* is `null`. -or-

	ptr is invalid.

1

2

Marshal.ReadInt16(System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static int16 ReadInt16(native int ptr, int32 ofs)
cil managed

[C#]
public static short ReadInt16 (IntPtr ptr, int ofs)
```

Summary

Reads a 16-bit signed integer at a given offset from unmanaged memory.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory from which to read.
ofs	An additional byte offset, which is added to the *ptr* parameter before reading.

Return Value

The 16-bit signed integer read from unmanaged memory at the given offset.

Description

`System.Runtime.InteropServices.Marshal.ReadInt16` enables direct interaction with an unmanaged 16-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.

Marshal.ReadInt16(System.Object, System.Int32) Method

```
[ILAsm]
.method public hidebysig static pinvokeimpl("mscoree.dll" as "ND_RI2"
winapi) int16 ReadInt16([in] object marshal(as any) ptr, int32 ofs) cil
managed

[C#]
public static short ReadInt16 (object ptr, int ofs)
```

Summary

Reads a 16-bit signed integer at a given offset from unmanaged memory.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory of the source object.
ofs	An additional byte offset, which is added to the *ptr* parameter before reading.

Return Value

The 16-bit signed integer read from unmanaged memory at the given offset.

Description

`System.Runtime.InteropServices.Marshal.ReadInt16` enables direct interaction with an unmanaged 16-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.
System.ArgumentException	*ptr* is an `System.Runtime.InteropServices.ArrayWithOffset` object. This method does not accept

	`System.Runtime.InteropServices.ArrayWithOffset` parameters.

1

2

Marshal.ReadInt32(System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static int32 ReadInt32(native int ptr) cil
managed

[C#]
public static int ReadInt32 (IntPtr ptr)
```

Summary

Reads a 32-bit signed integer from unmanaged memory.

Parameters

Parameter	Description
ptr	The address in unmanaged memory from which to read.

Return Value

The 32-bit signed integer read from unmanaged memory.

Description

`System.Runtime.InteropServices.Marshal.ReadInt32` has an implied offset of 0. This method enables direct interaction with an unmanaged C-style `Int32` array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	*ptr* is not a recognized format. -or- *ptr* is `null`. -or-

	ptr is invalid.

1

2

Marshal.ReadInt32(System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static int32 ReadInt32(native int ptr, int32 ofs)
cil managed

[C#]
public static int ReadInt32 (IntPtr ptr, int ofs)
```

Summary

Reads a 32-bit signed integer at a given offset from unmanaged memory.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory from which to read.
ofs	An additional byte offset, which is added to the *ptr* parameter before reading.

Return Value

The 32-bit signed integer read from unmanaged memory.

Description

`System.Runtime.InteropServices.Marshal.ReadInt32` enables direct interaction with an unmanaged 32-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.

Marshal.ReadInt32(System.Object, System.Int32) Method

```
[ILAsm]
.method public hidebysig static pinvokeimpl("mscoree.dllv as "ND_RI4"
winapi) int32 ReadInt32([in] object marshal(as any) ptr, int32 ofs) cil
managed

[C#]
public static int ReadInt32 (object ptr, int ofs)
```

Summary

Reads a 32-bit signed integer at a given offset from unmanaged memory.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory of the source object.
ofs	An additional byte offset, which is added to the *ptr* parameter before reading.

Return Value

The 32-bit signed integer read from unmanaged memory at the given offset.

Description

`System.Runtime.InteropServices.Marshal.ReadInt32` enables direct interaction with an unmanaged 32-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.
System.ArgumentException	*ptr* is an `System.Runtime.InteropServices.ArrayWithOffset` object. This method does not accept

	`System.Runtime.InteropServices.ArrayWithOffset` parameters.

1

2

Marshal.ReadInt64(System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static int64 ReadInt64(native int ptr) cil
managed

[C#]
public static long ReadInt64 (IntPtr ptr)
```

Summary

Reads a 64-bit signed integer from unmanaged memory.

Parameters

Parameter	Description
ptr	The address in unmanaged memory from which to read.

Return Value

The 64-bit signed integer read from unmanaged memory.

Description

`System.Runtime.InteropServices.Marshal.ReadInt64` has an implied offset of 0. This method enables direct interaction with an unmanaged C-style `Int64` array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	*ptr* is not a recognized format. -or- *ptr* is `null`. -or-

	ptr is invalid.

1

2

Marshal.ReadInt64(System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static int64 ReadInt64(native int ptr, int32 ofs)
cil managed

[C#]
public static long ReadInt64 (IntPtr ptr, int ofs)
```

Summary

Reads a 64-bit signed integer at a given offset from unmanaged memory.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory from which to read.
ofs	An additional byte offset, which is added to the *ptr* parameter before reading.

Return Value

The 64-bit signed integer read from unmanaged memory at the given offset.

Description

`System.Runtime.InteropServices.Marshal.ReadInt64` enables direct interaction with an unmanaged 64-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.

Marshal.ReadInt64(System.Object, System.Int32) Method

```
[ILAsm]
.method public hidebysig static pinvokeimpl("mscoree.dll" as "ND_RI8"
winapi) int64 ReadInt64([in] object marshal(as any) ptr, int32 ofs) cil
managed

[C#]
public static long ReadInt64 (object ptr, int ofs)
```

Summary

Reads a 64-bit signed integer at a given offset from unmanaged memory.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory of the source object.
ofs	An additional byte offset, which is added to the *ptr* parameter before reading.

Return Value

The 64-bit signed integer read from unmanaged memory at the given offset.

Description

`System.Runtime.InteropServices.Marshal.ReadInt64` enables direct interaction with an unmanaged 64-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.
System.ArgumentException	*ptr* is an `System.Runtime.InteropServices.ArrayWithOffset` object. This method does not accept

	`System.Runtime.InteropServices.ArrayWithOffset` parameters.

1

2

Marshal.ReadIntPtr(System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static native int ReadIntPtr(native int ptr) cil
managed

[C#]
public static IntPtr ReadIntPtr (IntPtr ptr)
```

Summary

Reads a processor native-sized integer from unmanaged memory.

Parameters

Parameter	Description
ptr	The address in unmanaged memory from which to read.

Return Value

The integer read from unmanaged memory. A 32 bit integer is returned on 32 bit machines and a 64 bit integer is returned on 64 bit machines.

Description

`System.Runtime.InteropServices.Marshal.ReadIntPtr` has an implied offset of 0. This method enables direct interaction with an unmanaged C-style `IntPtr` array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	*ptr* is not a recognized format. -or- *ptr* is `null`. -or-

	ptr is invalid.

1

2

Marshal.ReadIntPtr(System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static native int ReadIntPtr(native int ptr,
int32 ofs) cil managed

[C#]
public static IntPtr ReadIntPtr (IntPtr ptr, int ofs)
```

Summary

Reads a processor native sized integer at a given offset from unmanaged memory.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory from which to read.
ofs	An additional byte offset, which is added to the *ptr* parameter before reading.

Return Value

The integer read from unmanaged memory at the given offset.

Description

`System.Runtime.InteropServices.Marshal.ReadIntPtr` enables direct interaction with an unmanaged C-style `IntPtr` array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.

Marshal.ReadIntPtr(System.Object, System.Int32) Method

```
[ILAsm]
.method public hidebysig static native int ReadIntPtr([in] object
marshal(as any) ptr, int32 ofs) cil managed

[C#]
public static IntPtr ReadIntPtr (object ptr, int ofs)
```

Summary

Reads a processor native sized integer from unmanaged memory.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory of the source object.
ofs	An additional byte offset, which is added to the *ptr* parameter before reading.

Return Value

The integer read from unmanaged memory at the given offset.

Description

`System.Runtime.InteropServices.Marshal.ReadIntPtr` enables direct interaction with an unmanaged C-style `IntPtr` array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before reading its element values.

Reading from unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.
System.ArgumentException	*ptr* is an `System.Runtime.InteropServices.ArrayWithOffset` object. This method does not accept `System.Runtime.InteropServices.ArrayWithOffset`

	parameters.

1

2

Marshal.ReAllocHGlobal(System.IntPtr, System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static native int ReAllocHGlobal(native int pv,
native int cb) cil managed

[C#]
public static IntPtr ReAllocHGlobal (IntPtr pv, IntPtr cb)
```

Summary

Resizes a block of memory previously allocated with `System.Runtime.InteropServices.Marshal.AllocHGlobal`.

Parameters

Parameter	Description
pv	A pointer to memory allocated with `System.Runtime.InteropServices.Marshal. AllocHGlobal`.
cb	The new size of the allocated block.

Return Value

A pointer to the reallocated memory. This memory must be released using `System.Runtime.InteropServices.Marshal.FreeHGlobal`.

Description

`System.Runtime.InteropServices.Marshal.ReAllocHGlobal` is the memory allocation API method in the `System.Runtime.InteropServices.Marshal` class. The returned pointer can differ from the original.

Exceptions

Exception	Condition
System.OutOfMemoryException	There is insufficient memory to satisfy the request.

Marshal.SizeOf(System.Object) Method

```
[ILAsm]
.method public hidebysig static int32 SizeOf(object structure) cil managed

[C#]
public static int SizeOf (object structure)
```

Summary

Returns the unmanaged size of an object in bytes.

Parameters

Parameter	Description
structure	The object whose size is to be returned.

Return Value

The size of the specified object in unmanaged code.

Description

This method accepts an instance of a structure, which can be a reference type or a boxed value type. The layout must be sequential or explicit.

The size returned is the size of the unmanaged object. The unmanaged and managed sizes of an object can differ. For character types, the size is affected by the `System.Runtime.InteropServices.CharSet` value applied to that class.

You can use the `System.Runtime.InteropServices.Marshal.SizeOf` method to determine how much unmanaged memory to allocate using the `System.Runtime.InteropServices.Marshal.AllocHGlobal` method.

Exceptions

Exception	Condition
System.ArgumentNullException	The *structure* parameter is `null`.

Marshal.SizeOf(System.Type) Method

```
[ILAsm]
.method public hidebysig static int32 SizeOf(class System.Type t) cil
managed

[C#]
public static int SizeOf (Type t)
```

Summary

Returns the size of an unmanaged type in bytes.

Parameters

Parameter	Description
t	The type whose size is to be returned.

Return Value

The size of the specified type in unmanaged code.

Description

You can use this method when you do not have a structure. The layout must be sequential or explicit.

The size returned is the size of the unmanaged type. The unmanaged and managed sizes of an object can differ. For character types, the size is affected by the `System.Runtime.InteropServices.CharSet` value applied to that class.

Exceptions

Exception	Condition
System.ArgumentException	The *t* parameter is a generic type.
System.ArgumentNullException	The *t* parameter is `null`.

Marshal.StringToHGlobalAnsi(System.String) Method

```
[ILAsm]
.method public hidebysig static native int StringToHGlobalAnsi(string s)
cil managed

[C#]
public static IntPtr StringToHGlobalAnsi (string s)
```

Summary

Copies the contents of a managed `System.String` into unmanaged memory, converting into ANSI format as it copies.

Parameters

Parameter	Description
s	A managed string to be copied.

Return Value

The address, in unmanaged memory, to where *s* was copied, or 0 if *s* is `null`.

Description

`System.Runtime.InteropServices.Marshal.StringToHGlobalAnsi` is useful for custom marshaling or when mixing managed and unmanaged code. Because this method allocates the unmanaged memory required for a string, always free the memory by calling `System.Runtime.InteropServices.Marshal.FreeHGlobal`. `System.Runtime.InteropServices.Marshal.StringToHGlobalAnsi` provides the opposite functionality of `System.Runtime.InteropServices.Marshal.PtrToStringAnsi`.

Exceptions

Exception	Condition
System.OutOfMemoryException	There is insufficient memory available.
System.ArgumentOutOfRangeException	The *s* parameter exceeds the maximum length allowed by the operating system.

Marshal.StringToHGlobalAuto(System.String) Method

```
[ILAsm]
.method public hidebysig static native int StringToHGlobalAuto(string s)
cil managed

[C#]
public static IntPtr StringToHGlobalAuto (string s)
```

Summary

Copies the contents of a managed `System.String` into unmanaged memory, converting into ANSI format if required.

Parameters

Parameter	Description
s	A managed string to be copied.

Return Value

The address, in unmanaged memory, to where the string was copied, or 0 if *s* is `null`.

Description

`System.Runtime.InteropServices.Marshal.StringToHGlobalAuto` is useful for custom marshaling or for use when mixing managed and unmanaged code. Because this method allocates the unmanaged memory required for a string, always free the memory by calling `System.Runtime.InteropServices.Marshal.FreeHGlobal`. This method provides the opposite functionality of `System.Runtime.InteropServices.Marshal.PtrToStringAuto`.

Exceptions

Exception	Condition
System.OutOfMemoryException	There is insufficient memory available.

Marshal.StringToHGlobalUni(System.String) Method

```
[ILAsm]
.method public hidebysig static native int StringToHGlobalUni(string s)
cil managed

[C#]
public static IntPtr StringToHGlobalUni (string s)
```

Summary

Copies the contents of a managed `System.String` into unmanaged memory.

Parameters

Parameter	Description
s	A managed string to be copied.

Return Value

The address, in unmanaged memory, to where the *s* was copied, or 0 if *s* is `null`.

Description

`System.Runtime.InteropServices.Marshal.StringToHGlobalUni` is useful for custom marshaling or for use when mixing managed and unmanaged code. Because this method allocates the unmanaged memory required for a string, always free the memory by calling `System.Runtime.InteropServices.Marshal.FreeHGlobal`. This method provides the opposite functionality of `System.Runtime.InteropServices.Marshal.PtrToStringUni`.

Exceptions

Exception	Condition
System.OutOfMemoryException	The method could not allocate enough native heap memory.
System.ArgumentOutOfRangeException	The *s* parameter exceeds the maximum length allowed by the operating system.

Marshal.StructureToPtr(System.Object, System.IntPtr, System.Boolean) Method

```
[ILAsm]
.method public hidebysig static void StructureToPtr(object structure,
native int ptr, bool fDeleteOld) cil managed internalcall

[C#]
public static void StructureToPtr (object structure, IntPtr ptr, bool
fDeleteOld)
```

Summary

Marshals data from a managed object to an unmanaged block of memory.

Parameters

Parameter	Description
structure	A managed object holding the data to be marshaled. This object must be an instance of a formatted class.
ptr	A pointer to an unmanaged block of memory, which must be allocated before this method is called.
fDeleteOld	`true` to have the `System.Runtime.InteropServices.Marshal.DestroyStructure` method called on the *ptr* parameter before this method executes. Note that passing `false` can lead to a memory leak.

Description

`System.Runtime.InteropServices.Marshal.StructureToPtr` copies the contents of *structure* to the pre-allocated block of memory that the *ptr* parameter points to. If the *fDeleteOld* parameter is `true`, the pre-allocated buffer is deleted with the appropriate deletion method on the embedded pointer, but the buffer must contain valid data. This method cleans up every reference field specified in the mirrored managed class.

Suppose that *ptr* points to an unmanaged block of memory. The layout of this block is described by a corresponding managed class, which is specified by *structure*. `System.Runtime.InteropServices.Marshal.StructureToPtr` marshals field values from a structure to a pointer. Suppose the *ptr* block includes a reference field pointing to a string buffer that currently holds "abc", and the corresponding field on the managed side is a string that holds "vwxyz". If you do not specify otherwise, `System.Runtime.InteropServices.Marshal.StructureToPtr` allocates a new unmanaged buffer to hold "vwxyz", and hooks it up to the *ptr* block. This action casts the old buffer "abc" adrift without freeing it (its memory is not released back to the unmanaged heap). The result is an orphan buffer that represents a memory leak in your

code. If you set the *fDeleteOld* parameter to `true`, `System.Runtime.InteropServices.Marshal.StructureToPtr` frees the buffer that holds "abc" before allocating a new buffer for "vwxyz".

[*Note:* To pin an existing structure, instead of copying it, use the `System.Runtime.InteropServices.GCHandle` type to create a pinned handle for the structure.

]

Exceptions

Exception	Condition
System.ArgumentException	The *structure* parameter is a generic type.

Marshal.UnsafeAddrOfPinnedArrayElement(System.Array, System.Int32) Method

```
[ILAsm]
.method public hidebysig static native int
UnsafeAddrOfPinnedArrayElement(class System.Array arr, int32 index) cil
managed internalcall

[C#]
public static IntPtr UnsafeAddrOfPinnedArrayElement (Array arr, int index)
```

Summary

Gets the address of the element at the specified index inside the specified array.

Parameters

Parameter	Description
arr	The array that contains the desired element.
index	The index in the *arr* parameter of the desired element.

Return Value

The address of *index* inside *arr*.

Description

The array must be pinned using a `System.Runtime.InteropServices.GCHandle` before it is passed to this method. For maximum performance, this method does not validate the array passed to it; this can result in unexpected behavior.

Marshal.WriteByte(System.IntPtr, System.Byte) Method

```
[ILAsm]
.method public hidebysig static void WriteByte(native int ptr, uint8 val)
cil managed

[C#]
public static void WriteByte (IntPtr ptr, byte val)
```

Summary

Writes a single byte value to unmanaged memory.

Parameters

Parameter	Description
ptr	The address in unmanaged memory to write to.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteByte` enables direct interaction with an unmanaged C-style byte array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Exceptions

Exception	Condition
System.AccessViolationException	*ptr* is not a recognized format. -or- *ptr* is `null`. -or- *ptr* is invalid.

Marshal.WriteByte(System.IntPtr, System.Int32, System.Byte) Method

```
[ILAsm]
.method public hidebysig static void WriteByte(native int ptr, int32 ofs,
uint8 val) cil managed

[C#]
public static void WriteByte (IntPtr ptr, int ofs, byte val)
```

Summary

Writes a single byte value to unmanaged memory at a specified offset.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory to write to.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteByte` enables direct interaction with an unmanaged C-style byte array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.

Marshal.WriteByte(System.Object, System.Int32, System.Byte) Method

```
[ILAsm]
.method public hidebysig static pinvokeimpl("mscoree.dll" as "ND_WU1"
winapi) void WriteByte([in][out] object marshal(as any) ptr, int32 ofs,
uint8 val) cil managed

[C#]
public static void WriteByte (object ptr, int ofs, byte val)
```

Summary

Writes a single byte value to unmanaged memory at a specified offset.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory of the target object.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteByte` enables direct interaction with an unmanaged C-style byte array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.
System.ArgumentException	*ptr* is an `System.Runtime.InteropServices.ArrayWithOffset` object. This method does not accept `System.Runtime.InteropServices.ArrayWithOffset` parameters.

1

2

Marshal.WriteInt16(System.IntPtr, System.Char) Method

```
[ILAsm]
.method public hidebysig static void WriteInt16(native int ptr, char val)
cil managed

[C#]
public static void WriteInt16 (IntPtr ptr, char val)
```

Summary

Writes a character as a 16-bit integer value to unmanaged memory.

Parameters

Parameter	Description
ptr	The address in unmanaged memory to write to.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt16` enables direct interaction with an unmanaged 16-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	*ptr* is not a recognized format. -or- *ptr* is `null`. -or- *ptr* is invalid.

1

2

Marshal.WriteInt16(System.IntPtr, System.Int16) Method

```
[ILAsm]
.method public hidebysig static void WriteInt16(native int ptr, int16 val)
cil managed

[C#]
public static void WriteInt16 (IntPtr ptr, short val)
```

Summary

Writes a 16-bit integer value to unmanaged memory.

Parameters

Parameter	Description
ptr	The address in unmanaged memory to write to.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt16` enables direct interaction with an unmanaged 16-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	*ptr* is not a recognized format. -or- *ptr* is `null`. -or- *ptr* is invalid.

1

2

Marshal.WriteInt16(System.IntPtr, System.Int32, System.Char) Method

```
[ILAsm]
.method public hidebysig static void WriteInt16(native int ptr, int32 ofs,
char val) cil managed

[C#]
public static void WriteInt16 (IntPtr ptr, int ofs, char val)
```

Summary

Writes a 16-bit signed integer value to unmanaged memory at a specified offset.

Parameters

Parameter	Description
ptr	The base address in the native heap to write to.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt16` enables direct interaction with an unmanaged 16-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.

Marshal.WriteInt16(System.IntPtr, System.Int32, System.Int16) Method

```
[ILAsm]
.method public hidebysig static void WriteInt16(native int ptr, int32 ofs,
int16 val) cil managed

[C#]
public static void WriteInt16 (IntPtr ptr, int ofs, short val)
```

Summary

Writes a 16-bit signed integer value into unmanaged memory at a specified offset.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory to write to.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt16` enables direct interaction with an unmanaged 16-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.

Marshal.WriteInt16(System.Object, System.Int32, System.Char) Method

```
[ILAsm]
.method public hidebysig static void WriteInt16([in][out] object ptr,
int32 ofs, char val) cil managed

[C#]
public static void WriteInt16 (object ptr, int ofs, char val)
```

Summary

Writes a 16-bit signed integer value to unmanaged memory at a specified offset.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory of the target object.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt16` enables direct interaction with an unmanaged 16-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.
System.ArgumentException	*ptr* is an `System.Runtime.InteropServices.ArrayWithOffset` object. This method does not accept `System.Runtime.InteropServices.ArrayWithOffset`

	parameters.

1

2

Marshal.WriteInt16(System.Object, System.Int32, System.Int16) Method

```
[ILAsm]
.method public hidebysig static pinvokeimpl("mscoree.dll" as "ND_WI2"
winapi) void WriteInt16([in][out] object marshal(as any) ptr, int32 ofs,
int16 val) cil managed

[C#]
public static void WriteInt16 (object ptr, int ofs, short val)
```

Summary

Writes a 16-bit signed integer value to unmanaged memory at a specified offset.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory of the target object.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt16` enables direct interaction with an unmanaged 16-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.
System.ArgumentException	*ptr* is an `System.Runtime.InteropServices.ArrayWithOffset` object. This method does not accept `System.Runtime.InteropServices.ArrayWithOffset`

	parameters.

1

2

Marshal.WriteInt32(System.IntPtr, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void WriteInt32(native int ptr, int32 val)
cil managed

[C#]
public static void WriteInt32 (IntPtr ptr, int val)
```

Summary

Writes a 32-bit signed integer value to unmanaged memory.

Parameters

Parameter	Description
ptr	The address in unmanaged memory to write to.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt32` enables direct interaction with an unmanaged 32-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	*ptr* is not a recognized format. -or- *ptr* is `null`. -or- *ptr* is invalid.

1

2

Marshal.WriteInt32(System.IntPtr, System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static void WriteInt32(native int ptr, int32 ofs,
int32 val) cil managed

[C#]
public static void WriteInt32 (IntPtr ptr, int ofs, int val)
```

Summary

Writes a 32-bit signed integer value into unmanaged memory at a specified offset.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory to write to.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt32` enables direct interaction with an unmanaged 32-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.

Marshal.WriteInt32(System.Object, System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static pinvokeimpl("mscoree.dll" as "ND_WI4"
winapi) void WriteInt32([in][out] object marshal(as any) ptr, int32 ofs,
int32 val) cil managed

[C#]
public static void WriteInt32 (object ptr, int ofs, int val)
```

Summary

Writes a 32-bit signed integer value to unmanaged memory at a specified offset.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory of the target object.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt32` enables direct interaction with an unmanaged 32-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.
System.ArgumentException	*ptr* is an `System.Runtime.InteropServices.ArrayWithOffset` object. This method does not accept `System.Runtime.InteropServices.ArrayWithOffset`

	parameters.

1

2

Marshal.WriteInt64(System.IntPtr, System.Int64) Method

```
[ILAsm]
.method public hidebysig static void WriteInt64(native int ptr, int64 val)
cil managed

[C#]
public static void WriteInt64 (IntPtr ptr, long val)
```

Summary

Writes a 64-bit signed integer value to unmanaged memory.

Parameters

Parameter	Description
ptr	The address in unmanaged memory to write to.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt64` enables direct interaction with an unmanaged 64-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	*ptr* is not a recognized format. -or- *ptr* is `null`. -or- *ptr* is invalid.

1

2

Marshal.WriteInt64(System.IntPtr, System.Int32, System.Int64) Method

```
[ILAsm]
.method public hidebysig static void WriteInt64(native int ptr, int32 ofs,
int64 val) cil managed

[C#]
public static void WriteInt64 (IntPtr ptr, int ofs, long val)
```

Summary

Writes a 64-bit signed integer value to unmanaged memory at a specified offset.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory to write.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt64` enables direct interaction with an unmanaged 64-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.

Marshal.WriteInt64(System.Object, System.Int32, System.Int64) Method

```
[ILAsm]
.method public hidebysig static pinvokeimpl("mscoree.dll" as "ND_WI8"
winapi) void WriteInt64([in][out] object marshal(as any) ptr, int32 ofs,
int64 val) cil managed

[C#]
public static void WriteInt64 (object ptr, int ofs, long val)
```

Summary

Writes a 64-bit signed integer value to unmanaged memory at a specified offset.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory of the target object.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteInt64` enables direct interaction with an unmanaged 64-bit signed array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.
System.ArgumentException	*ptr* is an `System.Runtime.InteropServices.ArrayWithOffset` object. This method does not accept `System.Runtime.InteropServices.ArrayWithOffset`

	parameters.

1

2

Marshal.WriteIntPtr(System.IntPtr, System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static void WriteIntPtr(native int ptr, native
int val) cil managed

[C#]
public static void WriteIntPtr (IntPtr ptr, IntPtr val)
```

Summary

Writes a processor native sized integer value into unmanaged memory.

Parameters

Parameter	Description
ptr	The address in unmanaged memory to write to.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteIntPtr` enables direct interaction with an unmanaged C-style `IntPtr` array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	*ptr* is not a recognized format. -or- *ptr* is `null`. -or- *ptr* is invalid.

1

2

Marshal.WriteIntPtr(System.IntPtr, System.Int32, System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static void WriteIntPtr(native int ptr, int32
ofs, native int val) cil managed

[C#]
public static void WriteIntPtr (IntPtr ptr, int ofs, IntPtr val)
```

Summary

Writes a processor native-sized integer value to unmanaged memory at a specified offset.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory to write to.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

This method writes a 32 bit integer on 32 bit systems, and a 64 bit integer on 64 bit systems.

`System.Runtime.InteropServices.Marshal.WriteIntPtr` enables direct interaction with an unmanaged C-style `IntPtr` array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.

Marshal.WriteIntPtr(System.Object, System.Int32, System.IntPtr) Method

```
[ILAsm]
.method public hidebysig static void WriteIntPtr([in][out] object
marshal(as any) ptr, int32 ofs, native int val) cil managed

[C#]
public static void WriteIntPtr (object ptr, int ofs, IntPtr val)
```

Summary

Writes a processor native sized integer value to unmanaged memory.

Parameters

Parameter	Description
ptr	The base address in unmanaged memory of the target object.
ofs	An additional byte offset, which is added to the *ptr* parameter before writing.
val	The value to write.

Description

`System.Runtime.InteropServices.Marshal.WriteIntPtr` enables direct interaction with an unmanaged C-style byte array, eliminating the expense of copying an entire unmanaged array (using `System.Runtime.InteropServices.Marshal.Copy`) to a separate managed array before setting its element values.

Writing to unaligned memory locations is supported.

Exceptions

Exception	Condition
System.AccessViolationException	Base address (*ptr*) plus offset byte (*ofs*) produces a null or invalid address.
System.ArgumentException	*ptr* is an `System.Runtime.InteropServices.ArrayWithOffset` object. This method does not accept `System.Runtime.InteropServices.ArrayWithOffset`

	parameters.

1
2