System.Runtime.InteropServices.SafeBuffer Class

```
[ILAsm]
.class public abstract beforefieldinit SafeBuffer extends
System.Runtime.InteropServices.SafeHandle

[C#]
public abstract class SafeBuffer:
System.Runtime.InteropServices.SafeHandle
```

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 controlled memory buffer that can be used for reading and writing. Attempts to access memory outside the controlled buffer (underruns and overruns) raise exceptions.

Inherits From: System.Runtime.InteropServices.SafeHandle

Library: RuntimeInfrastructure

Description

You must call the `System.Runtime.InteropServices.SafeBuffer.Initialize` method before you use any instance of `System.Runtime.InteropServices.SafeBuffer`. To avoid races when you store an instance of a `System.Runtime.InteropServices.SafeBuffer` object in a static variable, you should use one of the following approaches:

- Create a lock when publishing the `System.Runtime.InteropServices.SafeBuffer`.
- Create a local variable, initialize the `System.Runtime.InteropServices.SafeBuffer`, and then assign the `System.Runtime.InteropServices.SafeBuffer` to the static variable, for example, by using the `System.Threading.Interlocked.CompareExchange``1` method.

[*Note:* Assignments in a static class constructor are implicitly locked.

]

SafeBuffer(System.Boolean) Constructor

```
[ILAsm]
.method family hidebysig specialname rtspecialname instance void
.ctor(bool ownsHandle) cil managed

[C#]
protected SafeBuffer (bool ownsHandle)
```

Summary

Creates a new instance of the `System.Runtime.InteropServices.SafeBuffer` class, and specifies whether the buffer handle is to be reliably released.

Parameters

Parameter	Description
ownsHandle	`true` to reliably release the handle during the finalization phase; `false` to prevent reliable release (not recommended).

SafeBuffer.AcquirePointer(System.Byte*&) Method

```
[ILAsm]
.method public hidebysig instance void AcquirePointer(uint8*& pointer) cil
managed

[C#]
public void AcquirePointer (ref byte* pointer)
```

Summary

Obtains a pointer from a `System.Runtime.InteropServices.SafeBuffer` object for a block of memory.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
pointer	A byte pointer, passed by reference, to receive the pointer from within the `System.Runtime.InteropServices.SafeBuffer` object. You must set this pointer to `null` before you call this method.

Description

When `System.Runtime.InteropServices.SafeBuffer.AcquirePointer` returns, you should perform bounds checking by verifying that the *pointer* parameter is `null`. If it is not `null`, you must call the `System.Runtime.InteropServices.SafeBuffer.ReleasePointer` method in a constrained execution region (CER).

`System.Runtime.InteropServices.SafeBuffer.AcquirePointer` calls the `System.Runtime.InteropServices.SafeHandle.DangerousAddRef` method and exposes the pointer. Unlike the `System.Runtime.InteropServices.SafeBuffer.Read``1` method, it does not change the current position of the pointer.

The following example demonstrates how to use the `System.Runtime.InteropServices.SafeBuffer.AcquirePointer` method:

```
byte* pointer = null;
RuntimeHelpers.PrepareConstrainedRegions();
try {
    MySafeBuffer.AcquirePointer(ref pointer);
```

```
    // Use pointer here, with your own bounds checking.
    }
finally {
    if (pointer != null)
        MySafeBuffer.ReleasePointer();
    }
```

If you cast *pointer* (which is a pointer to a byte) as a pointer to a different type (T*), you may have pointer alignment issues.

You must take responsibility for all bounds checking with this pointer.

Exceptions

Exception	Condition
System.InvalidOperationException	The `System.Runtime.InteropServices.SafeBuffer.Initialize` method has not been called.

SafeBuffer.Initialize(System.UInt64) Method

```
[ILAsm]
.method public hidebysig instance void Initialize(uint64 numBytes) cil
managed

[C#]
public void Initialize (ulong numBytes)
```

Summary

Defines the allocation size of the memory region in bytes. You must call this method before you use the `System.Runtime.InteropServices.SafeBuffer` instance.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
numBytes	The number of bytes in the buffer.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*numBytes* is less than zero. -or- *numBytes* is greater than the available address space.

SafeBuffer.Initialize(System.UInt32, System.UInt32) Method

```
[ILAsm]
.method public hidebysig instance void Initialize(uint32 numElements,
uint32 sizeOfEachElement) cil managed

[C#]
public void Initialize (uint numElements, uint sizeOfEachElement)
```

Summary

Specifies the allocation size of the memory buffer by using the specified number of elements and element size. You must call this method before you use the `System.Runtime.InteropServices.SafeBuffer` instance.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
numElements	The number of elements in the buffer.
sizeOfEachElement	The size of each element in the buffer.

Description

This method defines the required size of the memory region as the number of elements in an array multiplied by the size of each element.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*numElements* is less than zero. -or- *sizeOfEachElement* is less than zero. -or-

	numElements multiplied by *sizeOfEachElement* is greater than the available address space.

1

2

SafeBuffer.Initialize<T>(System.UInt32) Method

```
[ILAsm]
.method public hidebysig instance void Initialize<valuetype.ctor T>(uint32
numElements) cil managed

[C#]
public void Initialize<T> (uint numElements) where T: struct, new()
```

Summary

Defines the allocation size of the memory region by specifying the number of value types. You must call this method before you use the `System.Runtime.InteropServices.SafeBuffer` instance.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
numElements	The number of elements of the value type to allocate memory for.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*numElements* is less than zero. -or- *numElements* multiplied by the size of each element is greater than the available address space.

SafeBuffer.Read<T>(System.UInt64) Method

```
[ILAsm]
.method public hidebysig instance !!0 Read<valuetype.ctor T>(uint64
byteOffset) cil managed

[C#]
public T Read<T> (ulong byteOffset) where T: struct, new()
```

Summary

Reads a value type from memory at the specified offset.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
byteOffset	The location from which to read the value type. You may have to consider alignment issues.

Return Value

The value type that was read from memory.

Exceptions

Exception	Condition
System.InvalidOperationException	The `System.Runtime.InteropServices.SafeBuffer.Initialize` method has not been called.

SafeBuffer.ReadArray<T>(System.UInt64, T[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance void ReadArray<valuetype.ctor T>(uint64
byteOffset, !!0[] array, int32 index, int32 count) cil managed

[C#]
public void ReadArray<T> (ulong byteOffset, T[] array, int index, int
count) where T: struct, new()
```

Summary

Reads the specified number of value types from memory starting at the offset, and writes them into an array starting at the index.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
byteOffset	The location from which to start reading.
array	The output array to write to.
index	The location in the output array to begin writing to.
count	The number of value types to read from the input array and to write to the output array.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*index* is less than zero. -or- *count* is less than zero.

System.ArgumentNullException	*array* is `null`.
System.ArgumentException	The length of the array minus the index is less than *count*.
System.InvalidOperationException	The `System.Runtime.InteropServices.SafeBuffer.Initialize` method has not been called.

1

2

SafeBuffer.ReleasePointer() Method

```
[ILAsm]
.method public hidebysig instance void ReleasePointer() cil managed

[C#]
public void ReleasePointer ()
```

Summary

Releases a pointer that was obtained by the `System.Runtime.InteropServices.SafeBuffer.AcquirePointer` method.

Description

After this method returns, the pointer cannot be used.

Exceptions

Exception	Condition
System.InvalidOperationException	The `System.Runtime.InteropServices.SafeBuffer.Initialize` method has not been called.

SafeBuffer.Write<T>(System.UInt64, T) Method

```
[ILAsm]
.method public hidebysig instance void Write<valuetype.ctor T>(uint64
byteOffset, !!0 'value') cil managed

[C#]
public void Write<T> (ulong byteOffset, T value) where T: struct, new()
```

Summary

Writes a value type to memory at the given location.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
byteOffset	The location at which to start writing. You may have to consider alignment issues.
value	The value to write.

Description

This method is equivalent to the following code:

```
*(T*)(bytePtr + byteOffset) = value;
```

Exceptions

Exception	Condition
System.InvalidOperationException	The `System.Runtime.InteropServices.SafeBuffer.Initialize` method has not been called.

SafeBuffer.WriteArray<T>(System.UInt64, T[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance void WriteArray<valuetype.ctor T>(uint64
byteOffset, !!0[] array, int32 index, int32 count) cil managed

[C#]
public void WriteArray<T> (ulong byteOffset, T[] array, int index, int
count) where T: struct, new()
```

Summary

Writes the specified number of value types to a memory location by reading bytes starting from the specified location in the input array.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
byteOffset	The location in memory to write to.
array	The input array.
index	The offset in the array to start reading from.
count	The number of value types to write.

Description

Each element in the input array consists of the generic value type of the class.

Exceptions

Exception	Condition
System.ArgumentNullException	*array* is `null`.
System.ArgumentOutOfRangeException	*index* or *count* is less than zero.

System.ArgumentException	The length of the input array minus *index* is less than *count*.
System.InvalidOperationException	The `System.Runtime.InteropServices.SafeBuffer.Initialize` method has not been called.

SafeBuffer.ByteLength Property

```
[ILAsm]
.property instance uint64 ByteLength

[C#]
public ulong ByteLength { get; }
```

Summary

Gets the size of the buffer, in bytes.

Type Attributes:

- CLSCompliantAttribute(false)

Property Value

The number of bytes in the memory buffer.

Exceptions

Exception	Condition
System.InvalidOperationException	The `System.Runtime.InteropServices.SafeBuffer.Initialize` method has not been called.