

1 System.Runtime.InteropServices.SafeBuffer 2 Class

```
3 [ILAsm]  
4 .class public abstract beforefieldinit SafeBuffer extends  
5 System.Runtime.InteropServices.SafeHandle  
  
6 [C#]  
7 public abstract class SafeBuffer:  
8 System.Runtime.InteropServices.SafeHandle
```

9 Assembly Info:

- 10 • *Name:* mscorlib
- 11 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- 12 • *Version:* 4.0.0.0
- 13 • *Attributes:*
 - 14 ○ CLSCompliantAttribute(true)

15 Summary

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

19 Inherits From: System.Runtime.InteropServices.SafeHandle

20

21 **Library:** RuntimeInfrastructure

22

23 Description

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

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

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

35

36]

37

1 SafeBuffer(System.Boolean) Constructor

```
2 [ILAsm]  
3 .method family hidebysig specialname rtspecialname instance void  
4 .ctor(bool ownsHandle) cil managed  
  
5 [C#]  
6 protected SafeBuffer (bool ownsHandle)
```

7 Summary

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

10 Parameters

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

11

12

1 SafeBuffer.AcquirePointer(System.Byte* &) 2 Method

```
3 [ILAsm]  
4 .method public hidebysig instance void AcquirePointer(uint8*& pointer) cil  
5 managed  
6 [C#]  
7 public void AcquirePointer (ref byte* pointer)
```

8 Summary

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

11 Type Attributes:

- 12 • `CLSCompliantAttribute(false)`

13 Parameters

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

14 15 Description

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

21
22 `System.Runtime.InteropServices.SafeBuffer.AcquirePointer` calls the
23 `System.Runtime.InteropServices.SafeHandle.DangerousAddRef` method and
24 exposes the pointer. Unlike the
25 `System.Runtime.InteropServices.SafeBuffer.Read`1` method, it does not change
26 the current position of the pointer.

27
28 The following example demonstrates how to use the
29 `System.Runtime.InteropServices.SafeBuffer.AcquirePointer` method:

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

```
1     // Use pointer here, with your own bounds checking.
2     }
3 finally {
4     if (pointer != null)
5         MySafeBuffer.ReleasePointer();
6     }
```

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

9

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

11 **Exceptions**

Exception	Condition
System.InvalidOperationException	The <code>System.Runtime.InteropServices.SafeBuffer.Initialize</code> method has not been called.

12

13

1 SafeBuffer.Initialize(System.UInt64) Method

```
2 [ILAsm]  
3 .method public hidebysig instance void Initialize(uint64 numBytes) cil  
4 managed  
5 [C#]  
6 public void Initialize (ulong numBytes)
```

7 Summary

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

10 Type Attributes:

- 11 • `CLSCompliantAttribute(false)`

12 Parameters

Parameter	Description
<i>numBytes</i>	The number of bytes in the buffer.

13

14 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<i>numBytes</i> is less than zero. -or- <i>numBytes</i> is greater than the available address space.

15

16

1 SafeBuffer.Initialize(System.UInt32, 2 System.UInt32) Method

```
3 [ILAsm]  
4 .method public hidebysig instance void Initialize(uint32 numElements,  
5 uint32 sizeOfEachElement) cil managed  
  
6 [C#]  
7 public void Initialize (uint numElements, uint sizeOfEachElement)
```

8 Summary

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

12 Type Attributes:

- 13 • `CLSCompliantAttribute(false)`

14 Parameters

Parameter	Description
<i>numElements</i>	The number of elements in the buffer.
<i>sizeOfEachElement</i>	The size of each element in the buffer.

15 16 Description

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

19 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<i>numElements</i> is less than zero. -or- <i>sizeOfEachElement</i> is less than zero. -or-

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

1

2

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

2 Method

```
3 [ILAsm]  
4 .method public hidebysig instance void Initialize<valuetype.ctor T>(uint32  
5 numElements) cil managed  
  
6 [C#]  
7 public void Initialize<T> (uint numElements) where T: struct, new()
```

8 Summary

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

12 Type Attributes:

- 13 • `CLSCompliantAttribute(false)`

14 Parameters

Parameter	Description
<i>numElements</i>	The number of elements of the value type to allocate memory for.

15

16 Exceptions

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

17

18

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

```
2 [ILAsm]  
3 .method public hidebysig instance !!0 Read<valuetype.ctor T>(uint64  
4 byteOffset) cil managed  
  
5 [C#]  
6 public T Read<T>(ulong byteOffset) where T: struct, new()
```

7 Summary

8 Reads a value type from memory at the specified offset.

9 Type Attributes:

- 10 • CLSCompliantAttribute(false)

11 Parameters

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

12 Return Value

14 The value type that was read from memory.

15 Exceptions

Exception	Condition
System.InvalidOperationException	The <code>System.Runtime.InteropServices.SafeBuffer.Initialize</code> method has not been called.

16
17

1 SafeBuffer.ReadArray<T>(System.UInt64, 2 T[], System.Int32, System.Int32) Method

```
3 [ILAsm]  
4 .method public hidebysig instance void ReadArray<valuetype.ctor T>(uint64  
5 byteOffset, !!0[] array, int32 index, int32 count) cil managed  
  
6 [C#]  
7 public void ReadArray<T>(ulong byteOffset, T[] array, int index, int  
8 count) where T: struct, new()
```

9 Summary

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

12 Type Attributes:

- 13 • CLSCompliantAttribute(false)

14 Parameters

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

15

16 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<i>index</i> is less than zero. -or- <i>count</i> is less than zero.

System.ArgumentNullException	<i>array</i> is null.
System.ArgumentException	The length of the array minus the index is less than <i>count</i> .
System.InvalidOperationException	The <code>System.Runtime.InteropServices.SafeBuffer.Initialize</code> method has not been called.

1

2

1 SafeBuffer.ReleasePointer() Method

```
2 [ILAsm]  
3 .method public hidebysig instance void ReleasePointer() cil managed  
4 [C#]  
5 public void ReleasePointer ()
```

6 Summary

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

9 Description

10 After this method returns, the pointer cannot be used.

11 Exceptions

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

12

13

1 `SafeBuffer.Write<T>(System.UInt64, T)`

2 Method

```
3 [ILAsm]  
4 .method public hidebysig instance void Write<valuetype.ctor T>(uint64  
5 byteOffset, !!0 'value') cil managed  
  
6 [C#]  
7 public void Write<T>(ulong byteOffset, T value) where T: struct, new()
```

8 Summary

9 Writes a value type to memory at the given location.

10 Type Attributes:

- 11 • `CLSCompliantAttribute(false)`

12 Parameters

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

13 Description

15 This method is equivalent to the following code:

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

17 Exceptions

Exception	Condition
System.InvalidOperationException	The <code>System.Runtime.InteropServices.SafeBuffer.Initialize</code> method has not been called.

18

19

1 SafeBuffer.WriteAllArray<T>(System.UInt64, 2 T[], System.Int32, System.Int32) Method

```
3 [ILAsm]  
4 .method public hidebysig instance void WriteArray<valuetype.ctor T>(uint64  
5 byteOffset, !!0[] array, int32 index, int32 count) cil managed  
  
6 [C#]  
7 public void WriteArray<T> (ulong byteOffset, T[] array, int index, int  
8 count) where T: struct, new()
```

9 Summary

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

12 Type Attributes:

- 13 • CLSCompliantAttribute(false)

14 Parameters

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

15

16 Description

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

18 Exceptions

Exception	Condition
System.ArgumentNullException	<i>array</i> is null.
System.ArgumentOutOfRangeException	<i>index</i> or <i>count</i> is less than zero.

System.ArgumentException	The length of the input array minus <i>index</i> is less than <i>count</i> .
System.InvalidOperationException	The <code>System.Runtime.InteropServices.SafeBuffer.Initialize</code> method has not been called.

1

2

1 SafeBuffer.ByteLength Property

```
2 [ILAsm]  
3 .property instance uint64 ByteLength  
4 [C#]  
5 public ulong ByteLength { get; }
```

6 Summary

7 Gets the size of the buffer, in bytes.

8 Type Attributes:

- 9 • CLSCompliantAttribute(false)

10 Property Value

11 The number of bytes in the memory buffer.

12 Exceptions

Exception	Condition
System.InvalidOperationException	The <code>System.Runtime.InteropServices.SafeBuffer.Initialize</code> method has not been called.

13

14