System.Type Class

```
[ILAsm]
.class public abstract serializable Type extends System.Object

[C#]
public abstract class Type: Object
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 2.0.x.x
- *Attributes:*
  - CLSCompliantAttribute(true)

Summary

Provides information about a type.

Inherits From: System.Object [*Note:* When implementing the Reflection library, this type inherits from System.Reflection.MemberInfo.]

Library: BCL

Thread Safety: This type is safe for multithreaded operations.

Description

The `System.Type` class is abstract, as is the `System.Reflection.MemberInfo` class and its subclasses `System.Reflection.FieldInfo`, `System.Reflection.PropertyInfo`, `System.Reflection.MethodBase`, and `System.Reflection.EventInfo`. `System.Reflection.ConstructorInfo` and `System.Reflection.MethodInfo` are subclasses of `System.Reflection.MethodBase`. The runtime provides non-public implementations of these classes. [*Note:* For example, `System.Type.GetMethod` is typed as returning a `System.Reflection.MethodInfo` object. The returned object is actually an instance of the non-public runtime type that implements `System.Reflection.MethodInfo`.]

A conforming CLI program which is written to run on only the Kernel profile cannot subclass `System.Type`. [*Note:* This only applies to conforming programs not conforming implementations.]

A `System.Type` object that represents a type is unique; that is, two `System.Type` object references refer to the same object if and only if they represent the same type. This allows for comparison of `System.Type` objects using reference equality.

[*Note:* An instance of `System.Type` can represent any one of the following types:

- Classes
- Value types
- Arrays
- Interfaces
- Pointers
- Enumerations
- Constructed generic types and generic type definitions
- Type arguments and type parameters of constructed generic types, generic type definitions, and generic method definitions

The following table shows what members of a base class are returned by the methods that return members of types, such as `System.Type.GetConstructor` and `System.Type.GetMethod`.

Member Type	Static	Non-Static
Constructor	No	No
Field	No	Yes. A field is always hide-by-name-and-signature.
Event	Not applicable	The common type system rule is that the inheritance of an event is the same as that of the accessors that implement the event. Reflection treats events as hide-by-name-and-signature.
Method	No	Yes. A method (both virtual and non-virtual) can be hide-by-name or hide-by-name-and-signature.
Nested Type	No	No
Property	Not applicable	The common type system rule is that the inheritance is the same as that of the accessors that implement the property. Reflection treats properties as hide-by-name-and-signature.

For reflection, properties and events are hide-by-name-and-signature. If a property has both a get and a set accessor in the base class, but the derived class has only a get accessor, the derived class property hides the base class property, and the setter on the base class will not be accessible.

]

The description of `System.Type.IsGenericType` contains definitions for some important terms.

The following member must be implemented if the Reflection library is present in the implementation.

Type() Constructor

```
[ILAsm]
family rtspecialname specialname instance void .ctor()

[C#]
protected Type()
```

Summary

Constructs a new instance of the `System.Type` class.

The following member must be implemented if the Reflection library is present in the implementation.

Type.Delimiter Field

```
[ILAsm]
.field public static initOnly valuetype System.Char Delimiter

[C#]
public static readonly char Delimiter
```

Summary

Specifies the character that separates elements in the fully qualified name of a `System.Type`.

Description

This field is read-only.

The following member must be implemented if the Reflection library is present in the implementation.

Type.EmptyTypes Field

```
[ILAsm]
.field public static initOnly class System.Type[] EmptyTypes

[C#]
public static readonly Type[] EmptyTypes
```

Summary

Returns an empty array of type `System.Type`.

Description

This field is read-only.

The empty `System.Type` array returned by this field is used to specify that lookup methods in the `System.Type` class, such as `System.Type.GetMethod` and `System.Type.GetConstructor`, search for members that do not take parameters. [*Note:* For example, to locate the public instance constructor that takes no parameters, invoke `System.Type.GetConstructor` (`System.Reflection.BindingFlags.Public` | `System.Reflection.BindingFlags.Instance`, `null`, `System.Type.EmptyTypes`, `null`).]

The following member must be implemented if the Reflection library is present in the implementation.

Type.Missing Field

```
[ILAsm]
.field public static initOnly object Missing

[C#]
public static readonly object Missing
```

Summary

Represents a missing value in the `System.Type` information.

Description

This field is read-only.

Use the `Missing` field for invocation through reflection to ensure that a call will be made with the default value of a parameter as specified in the metadata. [*Note:* If the `Missing` field is specified for a parameter value and there is no default value for that parameter, a `System.ArgumentException` is thrown.]

The following member must be implemented if the Reflection library is present in the implementation.

Type.Equals(System.Type) Method

```
[ILAsm]
.method public hidebysig instance bool Equals(class System.Type o)

[C#]
public bool Equals(Type o)
```

Summary

Determines if the underlying system type of the current `System.Type` is the same as the underlying system type of the specified `System.Type`.

Parameters

Parameter	Description
o	The `System.Type` whose underlying system type is to be compared with the underlying system type of the current `System.Type`.

Return Value

`true` if the underlying system type of *o* is the same as the underlying system type of the current `System.Type`; otherwise, `false`.

Type.GetArrayRank() Method

```
[ILAsm]
.method public hidebysig virtual int32 GetArrayRank()

[C#]
public virtual int GetArrayRank()
```

Summary

Returns the number of dimensions in the current `System.Type`.

Return Value

A `System.Int32` containing the number of dimensions in the current `System.Type`.

Exceptions

Exception	Condition
System.ArgumentException	The current `System.Type` is not an array.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetAttributeFlagsImpl() Method

```
[ILAsm]
.method family hidebysig virtual abstract valuetype
System.Reflection.TypeAttributes GetAttributeFlagsImpl()

[C#]
protected abstract TypeAttributes GetAttributeFlagsImpl()
```

Summary

When overridden in a derived type implements the `System.Type.Attributes` property and returns the attributes specified for the type represented by the current instance.

Return Value

A `System.Reflection.TypeAttributes` value that signifies the attributes of the type represented by the current instance.

Behaviors

This property is read-only.

This method returns a `System.Reflection.TypeAttributes` value that indicates the attributes set in the metadata of the type represented by the current instance.

Usage

Use this property to determine the visibility, semantics, and layout format of the type represented by the current instance. Also use this property to determine if the type represented by the current instance has a special name.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetConstructor(System.Reflection.BindingFlags, System.Reflection.Binder, System.Type[], System.Reflection.ParameterModifier[]) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.ConstructorInfo
GetConstructor(valuetype System.Reflection.BindingFlags bindingAttr, class
System.Reflection.Binder binder, class System.Type[] types, class
System.Reflection.ParameterModifier[] modifiers)

[C#]
public ConstructorInfo GetConstructor(BindingFlags bindingAttr, Binder
binder, Type[] types, ParameterModifier[] modifiers)
```

Summary

Returns a constructor defined in the type represented by the current instance. The parameters of the constructor match the specified argument types and modifiers, under the specified binding constraints.

Parameters

Parameter	Description
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns null.
binder	A `System.Reflection.Binder` object that defines a set of properties and enables the binding, coercion of argument types, and invocation of members using reflection. Specify `null` to use the `System.Type.DefaultBinder`.
types	An array of `System.Type` objects. The elements in the array are of the same number, in the same order, and represent the same types as the parameters for the constructor to be returned.
modifiers	The only defined value for this parameter is `null`.

Return Value

A `System.Reflection.ConstructorInfo` object that reflects the constructor that matches the specified criteria. If an exact match does not exist, *binder* will attempt to coerce the parameter types specified in *types* to select a match. If *binder* is unable to select a match, returns `null`. If the type represented by the current instance is contained in a loaded assembly, the constructor that matches the specified criteria is not public, and the caller does not have sufficient permissions, returns `null`.

Description

The following `System.Reflection.BindingFlags` are used to define which constructors to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.
- Specify `System.Reflection.BindingFlags.Public` to include public constructors in the search.
- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public constructors (that is, private and protected constructors) in the search.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents a generic type, this method returns the `System.Reflection.ConstructorInfo` with the type parameters replaced by the appropriate type arguments. If the current instance represents an unassigned type parameter of a generic type or method, this method always returns `null`.

Exceptions

Exception	Condition
System.ArgumentNullException	*types* is `null`, or at least one of the elements in *types* is `null`.
System.ArgumentException	*types* has more than one dimension.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions`.

	ReflectionPermissionFlag.TypeInformation.

1

2

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetConstructor(System.Type[]) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.ConstructorInfo
GetConstructor(class System.Type[] types)

[C#]
public ConstructorInfo GetConstructor(Type[] types)
```

Summary

Returns a public instance constructor defined in the type represented by the current instance. The parameters of the constructor match the specified argument types.

Parameters

Parameter	Description
types	An array of `System.Type` objects. The elements in the array are of the same number, in the same order, and represent the same types as the parameters for the constructor to be returned. Specify `System.Type.EmptyTypes` to obtain a constructor that takes no parameters.

Return Value

A `System.Reflection.ConstructorInfo` object representing the public instance constructor whose parameters match exactly the types in *types*, if found; otherwise, `null`. If the type represented by the current instance is contained in a loaded assembly, the constructor that matches the specified criteria is not public, and the caller does not have sufficient permissions, returns `null`.

If the current instance represents a generic type, this method returns the `System.Reflection.ConstructorInfo` with the type parameters replaced by the appropriate type arguments. If the current instance represents an unassigned type parameter of a generic type or method, this method always returns `null`.

Description

This version of `System.Type.GetConstructor` is equivalent to `System.Type.GetConstructor(System.Reflection.BindingFlags.Public | System.Reflection.BindingFlags.Instance, null, types, null)`.

Exceptions

Exception	Condition
System.ArgumentNullException	*types* is `null`, or at least one of the elements in *types* is `null`.
System.ArgumentException	*types* has more than one dimension.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetConstructors() Method

```
[ILAsm]
.method public hidebysig instance class
System.Reflection.ConstructorInfo[] GetConstructors()

[C#]
public ConstructorInfo[] GetConstructors()
```

Summary

Returns an array of the public constructors defined in the type represented by the current instance.

Return Value

An array of `System.Reflection.ConstructorInfo` objects that reflect the public constructors defined in the type represented by the current instance. If no public constructors are defined in the type represented by the current instance, or if the current instance represents an unassigned type parameter of a generic type or method, returns an empty array.

If the current instance represents a generic type, this method returns the `System.Reflection.ConstructorInfo` objects with the type parameters replaced by the appropriate type arguments.

If the current instance represents a generic type parameter, the `System.Type.GetConstructors`method returns an empty array.

Description

This version of `System.Type.GetConstructors` is equivalent to `System.Type.GetConstructors(System.Reflection.BindingFlags.Public | System.Reflection.BindingFlags.Instance)`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetConstructors(System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public hidebysig virtual abstract class
System.Reflection.ConstructorInfo[] GetConstructors(valuetype
System.Reflection.BindingFlags bindingAttr)

[C#]
public abstract ConstructorInfo[] GetConstructors(BindingFlags
bindingAttr)
```

Summary

Returns an array of constructors defined in the type represented by the current instance, under the specified binding constraints.

Parameters

Parameter	Description
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.

Return Value

An array of `System.Reflection.ConstructorInfo` objects that reflect the constructors that are defined in the type represented by the current instance and match the constraints of *bindingAttr*. If `System.Reflection.BindingFlags.NonPublic` and `System.Reflection.BindingFlags.Static` are specified, this array includes the type initializer if it is defined. If no constructors meeting the constraints of *bindingAttr* are defined in the type represented by the current instance, or if the current instance represents an unassigned type parameter of a generic type or method, returns an empty array. If the type represented by the current instance is contained in a loaded assembly, the constructors that match the specified criteria are not public, and the caller does not have sufficient permission, returns `null`.

If the current instance represents a generic type, this method returns the `System.Reflection.ConstructorInfo` objects with the type parameters replaced by the appropriate type arguments.

If the current instance represents a generic type parameter, the `System.Type.GetConstructors`method returns an empty array.

Description

The following `System.Reflection.BindingFlags` are used to define which constructors to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.
- Specify `System.Reflection.BindingFlags.Public` to include public constructors in the search.
- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public constructors (that is, private and protected constructors) in the search.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

Behaviors

As described above.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetDefaultMembers() Method

```
[ILAsm]
.method public hidebysig virtual class System.Reflection.MemberInfo[]
GetDefaultMembers()

[C#]
public virtual MemberInfo[] GetDefaultMembers()
```

Summary

Returns an array of `System.Reflection.MemberInfo` objects that reflect the default members defined in the type represented by the current instance.

Return Value

An array of `System.Reflection.MemberInfo` objects reflecting the default members of the type represented by the current instance. If the type represented by the current instance does not have any default members, returns an empty array.

Description

If the current instance represents a generic type, this method returns the `System.Reflection.MemberInfo` objects with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the members of the class constraint, or the members of `System.Object` if there is no class constraint; the members of all interface constraints; and the members of any interfaces inherited from class or interface constraints.

Behaviors

The members returned by this method have the `System.Reflection.DefaultMemberAttribute` attribute.

Type.GetElementType() Method

```
[ILAsm]
.method public hidebysig virtual abstract class System.Type
GetElementType()

[C#]
public abstract Type GetElementType()
```

Summary

Returns the element type of the current `System.Type`.

Return Value

A `System.Type` that represents the type used to create the current instance if the current instance represents an array, pointer, or an argument passed by reference. Otherwise, returns `null` if the current instance is not an array or a pointer, or is not passed by reference, or represents a generic type or a type parameter of a generic type or method.

Example

The following example demonstrates the `System.Type.GetElementType` method.

[C#]

```
using System;
class TestType {
 public static void Main() {
 int[] array = {1,2,3};
 Type t = array.GetType();
 Type t2 = t.GetElementType();
 Console.WriteLine("{0} element type is {1}",array, t2.ToString());

 TestType newMe = new TestType();
 t = newMe.GetType();
 t2 = t.GetElementType();
 Console.WriteLine("{0} element type is {1}", newMe, t2==null? "null":
t2.ToString());
 }
}
```

The output is

```
System.Int32[] element type is System.Int32


TestType element type is null
```

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetEvent(System.String) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.EventInfo
GetEvent(string name)

[C#]
public EventInfo GetEvent(string name)
```

Summary

Returns a `System.Reflection.EventInfo` object reflecting the public event that has the specified name and is defined in the type represented by the current instance.

Parameters

Parameter	Description
name	A `System.String` containing the name of the public event to be returned.

Return Value

A `System.Reflection.EventInfo` object reflecting the public event that is named *name* and is defined in the type represented by the current instance, if found; otherwise, `null`.

If the current instance represents a generic type, this method returns the `System.Reflection.EventInfo` with the type parameters replaced by the appropriate type arguments.

Description

This version of `System.Type.GetEvent` is equivalent to `System.Type.GetEvent`(*name*, `System.Reflection.BindingFlags.Static` | `System.Reflection.BindingFlags.Instance` | `System.Reflection.BindingFlags.Public`).

The search for *name* is case-sensitive.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the events of the class constraint; the events of all interface constraints; and the events of any interfaces inherited from class or interface constraints.

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.

1

2

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetEvent(System.String, System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public hidebysig virtual abstract class
System.Reflection.EventInfo GetEvent(string name, valuetype
System.Reflection.BindingFlags bindingAttr)

[C#]
public abstract EventInfo GetEvent(string name, BindingFlags bindingAttr)
```

Summary

Returns a `System.Reflection.EventInfo` object reflecting the event that has the specified name, is defined in the type represented by the current instance, and matches the specified binding constraints.

Parameters

Parameter	Description
name	A `System.String` containing the name of the event to be returned.
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns null.

Return Value

A `System.Reflection.EventInfo` object reflecting the event that is named *name*, is defined in the type represented by the current instance, and matches the constraints of *bindingAttr*. If an event matching these criteria is not found, returns `null`. If the event is not public, the current instance represents a type from a loaded assembly, and the caller does not have sufficient permission, returns `null`.

If the current instance represents a generic type, this method returns the `System.Reflection.EventInfo` with the type parameters replaced by the appropriate type arguments.

Description

The following `System.Reflection.BindingFlags` are used to define which events to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.

- Specify `System.Reflection.BindingFlags.Public` to include public events in the search.

- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public events(that is, private and protected events) in the search.

The following `System.Reflection.BindingFlags` value can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the events declared on the type, not events that were simply inherited.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the events of the class constraint; the events of all interface constraints; and the events of any interfaces inherited from class or interface constraints.

Behaviors

As described above.

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetEvents() Method

```
[ILAsm]
.method public hidebysig virtual class System.Reflection.EventInfo[]
GetEvents()

[C#]
public virtual EventInfo[] GetEvents()
```

Summary

Returns an array of `System.Reflection.EventInfo` objects that reflect the public events defined in the type represented by the current instance.

Return Value

An array of `System.Reflection.EventInfo` objects that reflect the public events defined in the type represented by the current instance. If no public events are defined in the type represented by the current instance, returns an empty array.

If the current instance represents a generic type, this method returns the `System.Reflection.EventInfo` objects with the type parameters replaced by the appropriate type arguments.

Description

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the events of the class constraint; the events of all interface constraints; and the events of any interfaces inherited from class or interface constraints.

Behaviors

As described above.

Default

This version of `System.Type.GetEvents` is equivalent to `System.Type.GetEvents(System.Reflection.BindingFlags.Public | System.Reflection.BindingFlags.Static | System.Reflection.BindingFlags.Instance)`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetEvents(System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public hidebysig virtual abstract class
System.Reflection.EventInfo[] GetEvents(valuetype
System.Reflection.BindingFlags bindingAttr)

[C#]
public abstract EventInfo[] GetEvents(BindingFlags bindingAttr)
```

Summary

Returns an array of `System.Reflection.EventInfo` objects that reflect the events that are defined in the type represented by the current instance and match the specified binding constraints.

Parameters

Parameter	Description
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns null.

Return Value

An array of `System.Reflection.EventInfo` objects that reflect the events that are defined in the type represented by the current instance and match the constraints of *bindingAttr*. If no events match these constraints, returns an empty array. If the type reflected by the current instance is from a loaded assembly and the caller does not have permission to reflect on non-public objects in loaded assemblies, returns only public events.

If the current instance represents a generic type, this method returns the `System.Reflection.EventInfo` objects with the type parameters replaced by the appropriate type arguments.

Description

The following `System.Reflection.BindingFlags` are used to define which events to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.

- Specify `System.Reflection.BindingFlags.Public` to include public events in the search.

- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public events (that is, private and protected events) in the search.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the events of the class constraint; the events of all interface constraints; and the events of any interfaces inherited from class or interface constraints.

Behaviors

As described above.

Permissions

Permission	Description
System.Security.Permissions. ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions. ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetField(System.String, System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public hidebysig virtual abstract class
System.Reflection.FieldInfo GetField(string name, valuetype
System.Reflection.BindingFlags bindingAttr)

[C#]
public abstract FieldInfo GetField(string name, BindingFlags bindingAttr)
```

Summary

Returns a `System.Reflection.FieldInfo` object reflecting the field that has the specified name, is defined in the type represented by the current instance, and matches the specified binding constraints.

Parameters

Parameter	Description
name	A `System.String` containing the name of the field to be returned.
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.

Return Value

A `System.Reflection.FieldInfo` object reflecting the field that is named *name*, is defined in the type represented by the current instance, and matches the constraints of *bindingAttr*. If a field matching these criteria cannot be found, returns `null`. If the field is not public, the current type is from a loaded assembly, and the caller does not have sufficient permission, returns `null`.

If the current instance represents a generic type, this method returns the `System.Reflection.FieldInfo` with the type parameters replaced by the appropriate type arguments.

Description

The following `System.Reflection.BindingFlags` are used to define which fields to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.

- Specify `System.Reflection.BindingFlags.Public` to include public fields in the search.

- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public fields (that is, private and protected fields) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the fields declared in the type, not fields that were simply inherited.

- `System.Reflection.BindingFlags.IgnoreCase` to ignore the case of *name*.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the fields of the class constraint; the fields of all interface constraints; and the fields of any interfaces inherited from class or interface constraints.

Behaviors

As described above.

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetField(System.String) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.FieldInfo
GetField(string name)

[C#]
public FieldInfo GetField(string name)
```

Summary

Returns a `System.Reflection.FieldInfo` object reflecting the field that has the specified name and is defined in the type represented by the current instance.

Parameters

Parameter	Description
name	A `System.String` containing the name of the field to be returned.

Return Value

A `System.Reflection.FieldInfo` object reflecting the field that is named *name* and is defined in the type represented by the current instance, if found; otherwise, `null`. If the selected field is non-public, the type represented by the current instance is from a loaded assembly and the caller does not have sufficient permission to reflect on non-public objects in loaded assemblies, returns `null`.

If the current instance represents a generic type, this method returns the `System.Reflection.FieldInfo` with the type parameters replaced by the appropriate type arguments.

Description

The search for *name* is case-sensitive.

This version of `System.Type.GetField` is equivalent to `System.Type.GetField`(*name*, `System.Reflection.BindingFlags.Public` | `System.Reflection.BindingFlags.Static` | `System.Reflection.BindingFlags.Instance`).

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the fields of the class constraint; the fields of all interface constraints; and the fields of any interfaces inherited from class or interface constraints.

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetFields() Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.FieldInfo[]
GetFields()

[C#]
public FieldInfo[] GetFields()
```

Summary

Returns an array of `System.Reflection.FieldInfo` objects that reflect the public fields defined in the type represented by the current instance.

Return Value

An array of `System.Reflection.FieldInfo` objects that reflect the public fields defined in the type represented by the current instance. If no public fields are defined in the type represented by the current instance, returns an empty array.

If the current instance represents a generic type, this method returns the `System.Reflection.FieldInfo` objects with the type parameters replaced by the appropriate type arguments.

Description

This version of `System.Type.GetFields` is equivalent to `System.Type.GetFields`(`System.Reflection.BindingFlags.Instance` | `System.Reflection.BindingFlags.Static` | `System.Reflection.BindingFlags.Public`).

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the fields of the class constraint; the fields of all interface constraints; and the fields of any interfaces inherited from class or interface constraints.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetFields(System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public hidebysig virtual abstract class
System.Reflection.FieldInfo[] GetFields(valuetype
System.Reflection.BindingFlags bindingAttr)

[C#]
public abstract FieldInfo[] GetFields(BindingFlags bindingAttr)
```

Summary

Returns an array of `System.Reflection.FieldInfo` objects that reflect the fields that are defined in the type represented by the current instance and match the specified binding constraints.

Parameters

Parameter	Description
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.

Return Value

An array of `System.Reflection.FieldInfo` objects that reflect the fields that are defined in the type represented by the current instance and match the constraints of *bindingAttr*. If no fields match these constraints, returns an empty array. If the type represented by the current instance is from a loaded assembly and the caller does not have sufficient permission to reflect on non-public objects in loaded assemblies, returns only public fields.

If the current instance represents a generic type, this method returns the `System.Reflection.FieldInfo` objects with the type parameters replaced by the appropriate type arguments.

Description

The following `System.Reflection.BindingFlags` are used to define which fields to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` in order to get a return value other than `null`.

- Specify `System.Reflection.BindingFlags.Public` to include public fields in the search.

- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public fields (that is, private and protected fields) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the fields declared in the type, not fields that were simply inherited.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the fields of the class constraint; the fields of all interface constraints; and the fields of any interfaces inherited from class or interface constraints.

Behaviors

As described above.

Permissions

Permission	Description
System.Security.Permissions. ReflectionPermission	Requires permission to retrieve information on non-public members of a type in loaded assemblies. See `System.Security.Permissions. ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetGenericArguments() Method

```
[ILAsm]
.method public hidebysig virtual class System.Type[] GetGenericArguments()

[C#]
public virtual Type[] GetGenericArguments()
```

Summary

Returns an array of `System.Type` objects that represent the type arguments of a generic type or the type parameters of a generic type definition.

Return Value

An array of `System.Type` objects that represent the type arguments of a generic type or the type parameters of a generic type definition. Returns an empty array if the current type is not a generic type. The array elements are returned in the order in which they appear in the list of type arguments for the generic type.

Description

If the current type is a closed constructed type, the array returned by the `System.Type.GetGenericArguments` method contains the type arguments that are bound to the type parameters. If the current type is a generic type definition, the array contains the type parameters. If the current type is an open constructed type in which some of the type parameters are bound to specific types, the array contains both type arguments and type parameters.

For a list of the invariant conditions for terms used in generic reflection, see the `System.Type.IsGenericType` property description.

Example

For an example of using this method, see the example for `System.Type.GenericParameterPosition`.

Type.GetGenericParameterConstraints() Method

```
[ILAsm]
.method public hidebysig virtual class System.Type[]
GetGenericParameterConstraints()

[C#]
public virtual Type[] GetGenericParameterConstraints()
```

Summary

Returns an array of `System.Type` objects that represent the type constraints on the current generic type parameter.

Return Value

An array of `System.Type` objects that represent the type constraints on the current generic type parameter.

Description

Each constraint on a generic type parameter is expressed as a `System.Type` object. The first element of the array is the class constraint, if any. If a type parameter has no class constraint and no interface constraints, an empty array of `System.Type` is returned for that type parameter. Use `System.Reflection.GenericParameterAttributes` to get the special constraints.

For a list of the invariant conditions for terms used in generic reflection, see the `System.Type.IsGenericType` property description.

Exceptions

Exception	Condition
System.InvalidOperationException	The current `System.Type` object is not a generic type parameter. That is, the `System.Type.IsGenericParameter` property returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetGenericTypeDefinition() Method

```
[ILAsm]
.method public hidebysig virtual class System.Type
GetGenericTypeDefinition()

[C#]
public virtual Type GetGenericTypeDefinition()
```

Summary

Returns a `System.Type` object that represents a generic type from which the current type can be constructed.

Return Value

A `System.Type` object representing a generic type from which the current type can be constructed.

Description

If two constructed types are created from the same generic type definition, the `System.Type.GetGenericTypeDefinition` method returns the same `System.Type` object for both types.

If you call `System.Type.GetGenericTypeDefinition` on a `System.Type` object that already represents a generic type definition, it returns the current `System.Type`.

[*Note:* An array type whose element type is a generic type is not itself generic. Use `System.Type.IsGenericType` to determine whether a type is generic before calling `System.Type.GetGenericTypeDefinition`.]

For a list of the invariant conditions for terms used in generic reflection, see the `System.Type.IsGenericType` property description.

Exceptions

Exception	Condition
System.InvalidOperationException	The current type is not a generic type. That is, `System.Type.HasGenericArguments` returns `false`.

Example

For an example of using this method, see the example for `System.Type.MakeGenericType`.

Type.GetHashCode() Method

```
[ILAsm]
.method public hidebysig virtual int32 GetHashCode()

[C#]
public override int GetHashCode()
```

Summary

Generates a hash code for the current instance.

Return Value

A `System.Int32` containing the hash code for this instance.

Description

The algorithm used to generate the hash code is unspecified.

[*Note:* This method overrides `System.Object.GetHashCode`.]

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetInterface(System.String, System.Boolean) Method

```
[ILAsm]
.method public hidebysig virtual abstract class System.Type
GetInterface(string name, bool ignoreCase)

[C#]
public abstract Type GetInterface(string name, bool ignoreCase)
```

Summary

Returns the specified interface, specifying whether to do a case-sensitive search.

Parameters

Parameter	Description
name	A `System.String` containing the name of the interface to return.
ignoreCase	A `System.Boolean` where `true` indicates that the name search is to be done case-insensitively, and `false` performs a case-sensitive search.

Return Value

A `System.Type` object representing the interface with the specified name, implemented or inherited by the type represented by the instance, if found; otherwise, `null`.

Description

If the current instance represents a generic type, this method returns the `System.Type` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the interface constraints and any interfaces inherited from class or interface constraints.

[*Note:* The *name* parameter cannot include type arguments.]

[*Note:* Even with the introduction of generics, this method continues to return only non-generic members. To get the generic ones, one must call `System.Type.GetInterfaces`, and filter them out.]

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.
System.Reflection.AmbiguousMatchException	The current instance represents a type that implements the same generic interface with different type arguments.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetInterface(System.String) Method

```
[ILAsm]
.method public hidebysig instance class System.Type GetInterface(string
name)

[C#]
public Type GetInterface(string name)
```

Summary

Searches for the interface with the specified name.

Parameters

Parameter	Description
name	A `System.String` containing the name of the interface to get.

Return Value

A `System.Type` object representing the interface with the specified name, implemented or inherited by the current `System.Type`, if found; otherwise, `null`.

Description

The search for *name* is case-sensitive.

If the current instance represents a generic type, this method returns the `System.Type` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the interface constraints and any interfaces inherited from class or interface constraints.

[*Note:* The *name* parameter cannot include type arguments.]

[*Note:* Even with the introduction of generics, this method continues to return only non-generic members. To get the generic ones, one must call `System.Type.GetInterfaces`, and filter them out.]

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.
System.Reflection.AmbiguousMatchException	The current instance represents a type that implements the same generic interface with different type arguments.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetInterfaces() Method

```
[ILAsm]
.method public hidebysig virtual abstract class System.Type[]
GetInterfaces()

[C#]
public abstract Type[] GetInterfaces()
```

Summary

Returns all interfaces implemented or inherited by the type represented by the current instance.

Return Value

An array of `System.Type` objects representing the interfaces implemented or inherited by the type represented by the current instance. If no interfaces are found, returns an empty `System.Type` array.

Description

If the current instance represents a generic type, this method returns the `System.Type` objects with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the interface constraints and any interfaces inherited from class or interface constraints.

[*Note:* Even with the introduction of generics, the overloads of `System.Type.GetInterface` continue to return only non-generic members. To get the generic ones, one must call `System.Type.GetInterfaces`, and filter them out.]

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetMember(System.String, System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public hidebysig virtual class System.Reflection.MemberInfo[]
GetMember(string name, valuetype System.Reflection.BindingFlags
bindingAttr)

[C#]
public virtual MemberInfo[] GetMember(string name, BindingFlags
bindingAttr)
```

Summary

Returns an array of `System.Reflection.MemberInfo` objects that reflect the members defined in the type represented by the current instance that have the specified name and match the specified binding constraints.

Parameters

Parameter	Description
name	A `System.String` containing the name of the member to be returned.
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.

Return Value

An array of `System.Reflection.MemberInfo` objects that reflect the members named *name*, are defined in the type represented by the current instance and match the constraints of *bindingAttr*. If no members match these constraints, returns an empty array. If the selected member is non-public, the type reflected by the current instance is from a loaded assembly and the caller does not have sufficient permission to reflect on non-public objects in loaded assemblies, returns `null`.

Description

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.

- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.

- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.

- `System.Reflection.BindingFlags.IgnoreCase` to ignore the case of *name*.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents a generic type, this method returns the `System.Reflection.MemberInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the members of the class constraint, or the members of `System.Object` if there is no class constraint; the members of all interface constraints; and the members of any interfaces inherited from class or interface constraints.

[*Note:* The *name* parameter cannot include type arguments.]

Behaviors

As described above.

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

1

2

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetMember(System.String) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.MemberInfo[]
GetMember(string name)

[C#]
public MemberInfo[] GetMember(string name)
```

Summary

Returns an array of `System.Reflection.MemberInfo` objects that reflect the public members that have the specified name and are defined in the type represented by the current instance.

Parameters

Parameter	Description
name	A `System.String` containing the name of the members to be returned.

Return Value

An array of `System.Reflection.MemberInfo` objects that reflect the public members that are named *name* and are defined in the type represented by the current instance. If no public members with the specified name are defined in the type represented by the current instance, returns an empty array.

Description

This version of `System.Type.GetMember` is equivalent to `System.Type.GetMember`(*name*, `System.Reflection.BindingFlags.Static` | `System.Reflection.BindingFlags.Instance` | `System.Reflection.BindingFlags.Public`).

The search for *name* is case-sensitive.

If the current instance represents a generic type, this method returns the `System.Reflection.MemberInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the members of the class constraint, or the members of `System.Object` if there is no class constraint; the members of all interface constraints; and the members of any interfaces inherited from class or interface constraints.

[*Note:* The *name* parameter cannot include type arguments.]

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetMembers(System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public hidebysig virtual abstract class
System.Reflection.MemberInfo[] GetMembers(valuetype
System.Reflection.BindingFlags bindingAttr)

[C#]
public abstract MemberInfo[] GetMembers(BindingFlags bindingAttr)
```

Summary

Returns an array of `System.Reflection.MemberInfo` objects that reflect the members that are defined in the type represented by the current instance and match the specified binding constraints.

Parameters

Parameter	Description
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.

Return Value

An array of `System.Reflection.MemberInfo` objects that reflect the members defined in the type represented by the current instance that match the constraints of *bindingAttr*. If no members match these constraints, returns an empty array. If the type represented by the current instance is from a loaded assembly and the caller does not have sufficient permission to reflect on non-public objects in loaded assemblies, returns only public members.

Description

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.
- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.

- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents a generic type, this method returns the `System.Reflection.MemberInfo` objects with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the members of the class constraint, or the members of `System.Object` if there is no class constraint; the members of all interface constraints; and the members of any interfaces inherited from class or interface constraints.

Behaviors

As described above.

Permissions

Permission	Description
System.Security.Permissions. ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions. ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetMembers() Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.MemberInfo[]
GetMembers()

[C#]
public MemberInfo[] GetMembers()
```

Summary

Returns an array of `System.Reflection.MemberInfo` objects that reflect the public members defined in the type represented by the current instance.

Return Value

An array of `System.Reflection.MemberInfo` objects that reflect the public members defined in the type represented by the current instance. If no public members are defined in the type represented by the current instance, returns an empty array.

Description

This version of `System.Type.GetMembers` is equivalent to `System.Type.GetMembers(System.Reflection.BindingFlags.Public | System.Reflection.BindingFlags.Static | System.Reflection.BindingFlags.Instance)`.

If the current instance represents a generic type, this method returns the `System.Reflection.MemberInfo` objects with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the members of the class constraint, or the members of `System.Object` if there is no class constraint; the members of all interface constraints; and the members of any interfaces inherited from class or interface constraints.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetMethod(System.String, System.Reflection.BindingFlags, System.Reflection.Binder, System.Type[], System.Reflection.ParameterModifier[]) Method

```
[ILAsm]
.method public final hidebysig virtual class System.Reflection.MethodInfo
GetMethod(string name, valuetype System.Reflection.BindingFlags
bindingAttr, class System.Reflection.Binder binder, class System.Type[]
types, class System.Reflection.ParameterModifier[] modifiers)

[C#]
public MethodInfo GetMethod(string name, BindingFlags bindingAttr, Binder
binder, Type[] types, ParameterModifier[] modifiers)
```

Summary

Returns a `System.Reflection.MethodInfo` object that reflects the method that matches the specified criteria and is defined in the type represented by the current instance.

Parameters

Parameter	Description
name	A `System.String` containing the name of the method to be returned.
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.
binder	A `System.Reflection.Binder` object that defines a set of properties and enables the binding, coercion of argument types, and invocation of members using reflection. Specify `null` to use the `System.Type.DefaultBinder`.
types	An array of `System.Type` objects. The elements in the array are of the same number, in the same order, and represent the same types as the parameters for the method to be returned.
modifiers	The only defined value for this parameter is `null`.

Return Value

A `System.Reflection.MethodInfo` object that reflects the method defined in the type represented by the current instance that matches the specified criteria. If no method matching the specified criteria is found, returns `null`. If the selected method is non-public, the type reflected by the current instance is from a loaded assembly, and the caller does not have permission to reflect on non-public objects in loaded assemblies, returns `null`.

Description

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.
- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.
- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.
- `System.Reflection.BindingFlags.IgnoreCase` to ignore the case of *name*.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents a generic type, this method returns the `System.Reflection.MethodInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the methods of the class constraint, or the methods of `System.Object` if there is no class constraint; the methods of all interface constraints; and the methods of any interfaces inherited from class or interface constraints.

[*Note:* The *name* parameter cannot include type arguments.]

Exceptions

Exception	Condition
System.Reflection.AmbiguousMatchException	More than one method matching the specified criteria was found.
System.ArgumentNullException	*name* or *types* is `null`. -or- At least one of the elements in *types* is `null`.
System.ArgumentException	*types* has more than one dimension.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetMethod(System.String, System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public final hidebysig virtual class System.Reflection.MethodInfo
GetMethod(string name, valuetype System.Reflection.BindingFlags
bindingAttr)

[C#]
public MethodInfo GetMethod(string name, BindingFlags bindingAttr)
```

Summary

Returns a `System.Reflection.MethodInfo` object that reflects the method that has the specified name and is defined in the type represented by the current instance.

Parameters

Parameter	Description
name	A `System.String` containing the name of the method to be returned.
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.

Return Value

A `System.Reflection.MethodInfo` object that reflects the method that is defined in the type represented by the current instance and matches the specified criteria, if found; otherwise, `null`.

Description

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.
- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.
- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.
- `System.Reflection.BindingFlags.IgnoreCase` to ignore the case of *name*.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

This version of `System.Type.GetMethod` is equivalent to `System.Type.GetMethod`(*name*, *bindingAttr*, `null`, `null`, `null`).

If the current instance represents a generic type, this method returns the `System.Reflection.MethodInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the methods of the class constraint, or the methods of `System.Object` if there is no class constraint; the methods of all interface constraints; and the methods of any interfaces inherited from class or interface constraints.

[*Note:* The *name* parameter cannot include type arguments.]

Exceptions

Exception	Condition
System.Reflection.AmbiguousMatchException	More than one method matching the specified criteria was found.
System.ArgumentNullException	*name* is `null`.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions`.

	ReflectionPermissionFlag.TypeInformation.

1

2

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetMethod(System.String) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.MethodInfo
GetMethod(string name)

[C#]
public MethodInfo GetMethod(string name)
```

Summary

Returns a `System.Reflection.MethodInfo` object that reflects the public method that has the specified name and is defined in the type represented by the current instance.

Parameters

Parameter	Description
name	A `System.String` containing the name of the public method to be returned.

Return Value

A `System.Reflection.MethodInfo` object reflecting the public method that is defined in the type represented by the current instance and has the specified name, if found; otherwise, `null`.

Description

The search for *name* is case-sensitive.

This version of `System.Type.GetMethod` is equivalent to `System.Type.GetMethod`(*name*, `System.Reflection.BindingFlags.Public` | `System.Reflection.BindingFlags.Static` | `System.Reflection.BindingFlags.Instance`, `null`, `null`, `null`).

If the current instance represents a generic type, this method returns the `System.Reflection.MethodInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the methods of the class constraint, or the methods of `System.Object` if there is no class constraint; the methods of all interface constraints; and the methods of any interfaces inherited from class or interface constraints.

[*Note:* The *name* parameter cannot include type arguments.]

1
2

3 **Exceptions**

Exception	Condition
System.Reflection. AmbiguousMatchException	More than one method matching the specified criteria was found.
System.ArgumentNullException	*name* is `null`.

4

5

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetMethod(System.String, System.Type[]) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.MethodInfo
GetMethod(string name, class System.Type[] types)

[C#]
public MethodInfo GetMethod(string name, Type[] types)
```

Summary

Returns a `System.Reflection.MethodInfo` object that reflects the public method defined in the type represented by the current instance that has the specified name and parameter information.

Parameters

Parameter	Description
name	A `System.String` containing the name of the public method to be returned.
types	An array of `System.Type` objects. The elements in the array are of the same number, in the same order, and represent the same types as the parameters for the method to be returned.

Return Value

A `System.Reflection.MethodInfo` object reflecting the public method defined in the type represented by the current instance that matches the specified criteria. If no public method matching the specified criteria is found, returns `null`.

Description

The search for *name* is case-sensitive.

This version of `System.Type.GetMethod` is equivalent to `System.Type.GetMethod`(*name*, `System.Reflection.BindingFlags.Public` | `System.Reflection.BindingFlags.Static` | `System.Reflection.BindingFlags.Instance`, `null`, *types*, `null`).

If the current instance represents a generic type, this method returns the `System.Reflection.MethodInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the methods of the class constraint, or the methods of `System.Object` if there is no class constraint; the methods of all interface constraints; and the methods of any interfaces inherited from class or interface constraints.

[*Note:* The *name* parameter cannot include type arguments.]

Exceptions

Exception	Condition
System.Reflection.AmbiguousMatchException	More than one method matching the specified criteria was found.
System.ArgumentNullException	*name* or *types* is `null`. -or- At least one of the elements in *types* is `null`.
System.ArgumentException	*types* has more than one dimension.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetMethod(System.String, System.Type[], System.Reflection.ParameterModifier[]) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.MethodInfo
GetMethod(string name, class System.Type[] types, class
System.Reflection.ParameterModifier[] modifiers)

[C#]
public MethodInfo GetMethod(string name, Type[] types, ParameterModifier[]
modifiers)
```

Summary

Returns a `System.Reflection.MethodInfo` object that reflects the public method that has the specified name and is defined in the type represented by the current instance.

Parameters

Parameter	Description
name	A `System.String` containing the name of the public method to be returned.
types	An array of `System.Type` objects. The elements in the array are of the same number, in the same order, and represent the same types as the parameters for the method to be returned.
modifiers	The only defined value for this parameter is `null`.

Return Value

A `System.Reflection.MethodInfo` object reflecting the public method that is defined in the type represented by the current instance and matches the specified criteria, if found; otherwise, `null`.

Description

The default binder does not process *modifier*.

The search for *name* is case-sensitive.

This version of `System.Type.GetMethod` is equivalent to `System.Type.GetMethod` (*name*, `System.Reflection.BindingFlags.Public` |`System.Reflection.BindingFlags.Static` |`System.Reflection.BindingFlags.Instance`, `null`, *types*, *modifiers*).

If the current instance represents a generic type, this method returns the `System.Reflection.MethodInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the methods of the class constraint, or the methods of `System.Object` if there is no class constraint; the methods of all interface constraints; and the methods of any interfaces inherited from class or interface constraints.

[*Note:* The *name* parameter cannot include type arguments.]

Exceptions

Exception	Condition
System.Reflection. AmbiguousMatchException	More than one method matching the specified criteria was found.
System.ArgumentNullException	*name* or *types* is `null`. -or- At least one of the elements in *types* is `null`.
System.ArgumentException	*types* has more than one dimension.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetMethods(System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public hidebysig virtual abstract class
System.Reflection.MethodInfo[] GetMethods(valuetype
System.Reflection.BindingFlags bindingAttr)

[C#]
public abstract MethodInfo[] GetMethods(BindingFlags bindingAttr)
```

Summary

Returns an array of `System.Reflection.MethodInfo` objects that reflect the methods defined in the type represented by the current instance that match the specified binding constraints.

Parameters

Parameter	Description
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.

Return Value

An array of `System.Reflection.MethodInfo` objects reflecting the methods defined in the type represented by the current instance that match the constraints of *bindingAttr*. If no such methods found, returns an empty array. If the type represented by the current instance is from a loaded assembly and the caller does not have permission to reflect on non-public objects in loaded assemblies, returns only public methods.

Description

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.
- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.
- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents a generic type, this method returns the `System.Reflection.MethodInfo` objects with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the methods of the class constraint, or the methods of `System.Object` if there is no class constraint; the methods of all interface constraints; and the methods of any interfaces inherited from class or interface constraints.

Behaviors

As described above.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetMethods() Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.MethodInfo[]
GetMethods()

[C#]
public MethodInfo[] GetMethods()
```

Summary

Returns the public methods defined in the type represented by the current instance.

Return Value

An array of `System.Reflection.MethodInfo` objects reflecting the public methods defined in the type represented by the current instance. If no methods are found, returns an empty array.

Description

This version of `System.Type.GetMethods` is equivalent to `System.Type.GetMethods(System.Reflection.BindingFlags.Instance | System.Reflection.BindingFlags.Static | System.Reflection.BindingFlags.Public )`.

If the current instance represents a generic type, this method returns the `System.Reflection.MethodInfo` objects with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the methods of the class constraint, or the methods of `System.Object` if there is no class constraint; the methods of all interface constraints; and the methods of any interfaces inherited from class or interface constraints.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetNestedType(System.String) Method

```
[ILAsm]
.method public hidebysig instance class System.Type GetNestedType(string
name)

[C#]
public Type GetNestedType(string name)
```

Summary

Returns the public nested type defined in the type represented by the current instance

Parameters

Parameter	Description
name	A `System.String` containing the name of the public nested type to return. Specify the unqualified name of the nested type. [*Note:* For example, for a type B nested within A, if typeA represents the type object for A, the correct invocation is typeA.GetNestedType("B").]

Return Value

A `System.Type` object representing the public nested type with the specified name, if found; otherwise, `null`.

Description

The search for *name* is case-sensitive.

Use the simple name of the nested class for *name*; do not qualify it with the name of the outer class. CLS rules require a naming pattern for nested types; see Partition I.

If the current instance represents an unassigned type parameter of a generic type or method definition, this method does not search the nested types of the class constraint.

[*Note:* The *name* parameter cannot include type arguments. For example, passing "MyGenericNestedType<int>" to this method searches for a nested type with the text name "MyGenericNestedType<int>", rather than for a nested type named MyGenericNestedType that has one generic argument of type `int`.]

[*Note:* If the nested type is generic, what is returned is always a generic type definition.]

For information on constructing nested generic types from their generic type definitions, see the `System.Type.MakeGenericType(System.Type[])` method.

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetNestedType(System.String, System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public hidebysig virtual abstract class System.Type
GetNestedType(string name, valuetype System.Reflection.BindingFlags
bindingAttr)

[C#]
public abstract Type GetNestedType(string name, BindingFlags bindingAttr)
```

Summary

Returns a nested types defined in the type represented by the current instance that match the specified binding constraints.

Parameters

Parameter	Description
name	A `System.String` containing the name of the nested type to return. Specify the unqualified name of the nested type. [*Note:* For example, for a type B nested within A, if typeA represents the type object for A, the correct invocation is typeA.GetNestedType("B").]
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.

Return Value

A `System.Type` object representing the nested type that matches the specified criteria, if found; otherwise, `null`. If the selected nested type is non-public, the current instance represents a type contained in a loaded assembly and the caller does not have sufficient permissions, returns `null`.

Description

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.

- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.

- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.

- `System.Reflection.BindingFlags.IgnoreCase` to ignore the case of *name*.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the nested types of the class constraint.

[*Note:* The *name* parameter cannot include type arguments.]

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetNestedTypes() Method

```
[ILAsm]
.method public hidebysig instance class System.Type[] GetNestedTypes()

[C#]
public Type[] GetNestedTypes()
```

Summary

Returns all the public types nested within the current `System.Type`.

Return Value

An array of `System.Type` objects representing all public types nested within the type represented by the current instance, if any. Otherwise, returns an empty `System.Type` array.

Description

This version of `System.Type.GetNestedTypes` is equivalent to `System.Type.GetNestedTypes(System.Reflection.BindingFlags.Public)`.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the nested types of the class constraint.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetNestedTypes(System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public hidebysig virtual abstract class System.Type[]
GetNestedTypes(valuetype System.Reflection.BindingFlags bindingAttr)

[C#]
public abstract Type[] GetNestedTypes(BindingFlags bindingAttr)
```

Summary

Returns an array containing the nested types defined in the type represented by the current instance that match the specified binding constraints.

Parameters

Parameter	Description
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.

Return Value

An array of `System.Type` objects representing all types nested within the type represented by the current instance that match the specified binding constraints, if any. Otherwise, returns an empty `System.Type` array. If the type reflected by the current instance is contained in a loaded assembly, the type that matches the specified criteria is not public, and the caller does not have sufficient permission, returns only public types.

Description

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.
- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the nested types of the class constraint.

Permissions

Permission	Description
System.Security.Permissions. ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions. ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetProperties(System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public hidebysig virtual abstract class
System.Reflection.PropertyInfo[] GetProperties(valuetype
System.Reflection.BindingFlags bindingAttr)

[C#]
public abstract PropertyInfo[] GetProperties(BindingFlags bindingAttr)
```

Summary

Returns an array of `System.Reflection.PropertyInfo` objects that reflect the properties defined for the type represented by the current instance that match the specified binding constraints.

Parameters

Parameter	Description
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.

Return Value

An array of `System.Reflection.PropertyInfo` objects that reflect the properties defined in the type represented by the current instance and match the constraints of *bindingAttr*. If no matching properties are found, returns an empty array. If the type represented by the current instance is from a loaded assembly and the caller does not have permission to reflect on non-public objects in loaded assemblies, returns only public properties.

Description

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.
- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.

- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

If the current instance represents a generic type, this method returns the `System.Reflection.PropertyInfo` objects with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the properties of the class constraint; the properties of all interface constraints; and the properties of any interfaces inherited from class or interface constraints.

Behaviors

A property is considered by reflection to be `public` if it has at least one accessor that is `public`. Otherwise, the property is not `public`.

Permissions

Permission	Description
System.Security.Permissions. ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions. ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetProperties() Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.PropertyInfo[]
GetProperties()

[C#]
public PropertyInfo[] GetProperties()
```

Summary

Returns an array of `System.Reflection.PropertyInfo` objects that reflect the public properties defined in the type represented by the current instance.

Return Value

An array of `System.Reflection.PropertyInfo` objects that reflect the public properties defined in the type represented by the current instance. If no public properties are found, returns an empty array.

Description

This version of `System.Type.GetProperties` is equivalent to `System.Type.GetProperties( System.Reflection.BindingFlags.Instance | System.Reflection.BindingFlags.Static | System.Reflection.BindingFlags.Public )`.

A property is considered by reflection to be `public` if it has at least one accessor that is `public`. Otherwise, the property is considered to be not `public`.

If the current instance represents a generic type, this method returns the `System.Reflection.PropertyInfo` objects with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the properties of the class constraint; the properties of all interface constraints; and the properties of any interfaces inherited from class or interface constraints.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetProperty(System.String, System.Type, System.Type[]) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.PropertyInfo
GetProperty(string name, class System.Type returnType, class System.Type[]
types)

[C#]
public PropertyInfo GetProperty(string name, Type returnType, Type[]
types)
```

Summary

Returns a `System.Reflection.PropertyInfo` object that reflects the public property defined in the type represented by the current instance that matches the specified search criteria.

Parameters

Parameter	Description
name	A `System.String` containing the name of the public property to be returned.
returnType	A `System.Type` object that represents the type of the public property to be returned.
types	An array of `System.Type` objects. The elements in the array are of the same number, in the same order, and represent the same types as the parameters for the indexer to be returned. Specify `System.Type.EmptyTypes` for a property that is not indexed.

Return Value

A `System.Reflection.PropertyInfo` object reflecting the public property defined in the type represented by the current instance that matches the specified criteria. If no matching property is found, returns `null`.

Description

This version of `System.Type.GetProperty` is equivalent to `System.Type.GetPropertyImpl`(*name*, `System.Reflection.BindingFlags.Static` | `System.Reflection.BindingFlags.Instance` | `System.Reflection.BindingFlags.Public`, `null`, *returnTypes*, *types*, `null`).

The search for *name* is case-sensitive.

Different programming languages use different syntax to specify indexed properties. Internally, this property is referred to by the name "Item" in the metadata. Therefore, any attempt to retrieve an indexed property using reflection is required to specify this internal name in order for the `PropertyInfo` to be returned correctly.

If the current instance represents a generic type, this method returns the `System.Reflection.PropertyInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the properties of the class constraint; the properties of all interface constraints; and the properties of any interfaces inherited from class or interface constraints.

Exceptions

Exception	Condition
System.Reflection.AmbiguousMatchException	More than one property matching the specified criteria was found.
System.ArgumentNullException	*name* or *types* is `null`, or at least one of the elements in *types* is `null`.
System.ArgumentException	*types* has more than one dimension.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetProperty(System.String, System.Type[]) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.PropertyInfo
GetProperty(string name, class System.Type[] types)

[C#]
public PropertyInfo GetProperty(string name, Type[] types)
```

Summary

Returns a `System.Reflection.PropertyInfo` object that reflects the public property defined in the type represented by the current instance that matches the specified search criteria.

Parameters

Parameter	Description
name	A `System.String` containing the name of the public property to be returned.
types	An array of `System.Type` objects. The elements in the array are of the same number, in the same order, and represent the same types as the parameters for the indexer to be returned. Specify `System.Type.EmptyTypes` to obtain a property that is not indexed.

Return Value

A `System.Reflection.PropertyInfo` object reflecting the public property defined on the type represented by the current instance that matches the specified criteria. If no matching property is found, returns `null`.

Description

This version of `System.Type.GetProperty` is equivalent to `System.Type.GetPropertyImpl`(*name*, `System.Reflection.BindingFlags.Static` | `System.Reflection.BindingFlags.Instance` | `System.Reflection.BindingFlags.Public`, `null`, `null`, *types*, `null`).

The search for *name* is case-sensitive.

Different programming languages use different syntax to specify indexed properties. Internally, this property is referred to by the name "Item" in the metadata. Therefore, any attempt to retrieve an indexed property using reflection is required to specify this

internal name in order for the `PropertyInfo` to be returned correctly.

If the current instance represents a generic type, this method returns the `System.Reflection.PropertyInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the properties of the class constraint; the properties of all interface constraints; and the properties of any interfaces inherited from class or interface constraints.

Exceptions

Exception	Condition
System.Reflection. AmbiguousMatchException	More than one property matching the specified criteria was found.
System.ArgumentNullException	*name* or *types* is `null`, or at least one of the elements in *types* is `null`.
System.ArgumentException	*types* has more than one dimension.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetProperty(System.String, System.Type) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.PropertyInfo
GetProperty(string name, class System.Type returnType)

[C#]
public PropertyInfo GetProperty(string name, Type returnType)
```

Summary

Returns a `System.Reflection.PropertyInfo` object that reflects the public property defined in the type represented by the current instance that matches the specified search criteria.

Parameters

Parameter	Description
name	A `System.String` containing the name of the property to be returned.
returnType	A `System.Type` object that represents the type of the property to be returned.

Return Value

A `System.Reflection.PropertyInfo` object reflecting the public property defined on the type represented by the current instance that matches the specified criteria. If no matching property is found, returns `null`.

Description

This version of `System.Type.GetProperty` is equivalent to `System.Type.GetPropertyImpl`(*name*, `System.Reflection.BindingFlags.Static` | `System.Reflection.BindingFlags.Instance` | `System.Reflection.BindingFlags.Public`, `null`, *returnType*, `null`, `null`).

The search for *name* is case-sensitive.

If the current instance represents a generic type, this method returns the `System.Reflection.PropertyInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the properties of the class constraint; the properties of all

interface constraints; and the properties of any interfaces inherited from class or interface constraints.

Exceptions

Exception	Condition
System.Reflection.AmbiguousMatchException	More than one property matching the specified criteria was found.
System.ArgumentNullException	*name* is `null`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetProperty(System.String) Method

```
[ILAsm]
.method public hidebysig instance class System.Reflection.PropertyInfo
GetProperty(string name)

[C#]
public PropertyInfo GetProperty(string name)
```

Summary

Returns a `System.Reflection.PropertyInfo` object that reflects the public property defined in the type represented by the current instance that has the specified name.

Parameters

Parameter	Description
name	A `System.String` containing the name of the property to be returned.

Return Value

A `System.Reflection.PropertyInfo` object reflecting the public property defined on the type represented by the current instance that has the specified name. If no matching property is found, returns `null`.

Description

This version of `System.Type.GetProperty` is equivalent to `System.Type.GetPropertyImpl`(*name*, `System.Reflection.BindingFlags.Static` | `System.Reflection.BindingFlags.Instance` | `System.Reflection.BindingFlags.Public`, `null`, `null`, `null`, `null`).

The search for *name* is case-sensitive.

If the current instance represents a generic type, this method returns the `System.Reflection.PropertyInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the properties of the class constraint; the properties of all interface constraints; and the properties of any interfaces inherited from class or interface constraints.

Exceptions

Exception	Condition
System.Reflection.AmbiguousMatchException	More than one property matching the specified criteria was found.
System.ArgumentNullException	*name* is `null`.

1

2

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetProperty(System.String, System.Reflection.BindingFlags) Method

```
[ILAsm]
.method public final hidebysig virtual class
System.Reflection.PropertyInfo GetProperty(string name, valuetype
System.Reflection.BindingFlags bindingAttr)

[C#]
public PropertyInfo GetProperty(string name, BindingFlags bindingAttr)
```

Summary

Returns a `System.Reflection.PropertyInfo` object that reflects the property defined in the type represented by the current instance that matches the specified search criteria.

Parameters

Parameter	Description
name	A `System.String` containing the name of the property to be returned.
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.

Return Value

A `System.Reflection.PropertyInfo` object reflecting the property defined in the type represented by the current instance that matches the specified criteria. If no matching property is found, returns `null`. If the type reflected by the current instance is contained in a loaded assembly, the property that matches the specified criteria is not public, and the caller does not have sufficient permission, returns `null`.

Description

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.
- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.

- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.

- `System.Reflection.BindingFlags.IgnoreCase` to ignore the case of *name*.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

This version of `System.Type.GetProperty` is equivalent to `System.Type.GetPropertyImpl`(*name*, *bindingAttr*, `null`, `null`, `null`, `null`).

The search for *name* is case-sensitive.

If the current instance represents a generic type, this method returns the `System.Reflection.PropertyInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the properties of the class constraint; the properties of all interface constraints; and the properties of any interfaces inherited from class or interface constraints.

Exceptions

Exception	Condition
System.Reflection.AmbiguousMatchException	More than one property matching the specified criteria was found.
System.ArgumentNullException	*name* is `null`.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions`.

	ReflectionPermissionFlag.TypeInformation.

1

2

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetProperty(System.String, System.Reflection.BindingFlags, System.Reflection.Binder, System.Type, System.Type[], System.Reflection.ParameterModifier[]) Method

```
[ILAsm]
.method public final hidebysig virtual class
System.Reflection.PropertyInfo GetProperty(string name, valuetype
System.Reflection.BindingFlags bindingAttr, class System.Reflection.Binder
binder, class System.Type returnType, class System.Type[] types, class
System.Reflection.ParameterModifier[] modifiers)

[C#]
public PropertyInfo GetProperty(string name, BindingFlags bindingAttr,
Binder binder, Type returnType, Type[] types, ParameterModifier[]
modifiers)
```

Summary

Returns a `System.Reflection.PropertyInfo` object that reflects the property defined in the type represented by the current instance that matches the specified search criteria.

Parameters

Parameter	Description
name	A `System.String` containing the name of the property to be returned.
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.
binder	A `System.Reflection.Binder` object that defines a set of properties and enables the binding, coercion of argument types, and invocation of members using reflection. Specify `null` to use the `System.Type.DefaultBinder`.
returnType	A `System.Type` object that represents the type of the property to be returned.
types	An array of `System.Type` objects. The elements in the array are of the same number, in the same order, and represent the same types as the parameters

	for the indexer to be returned. Specify `System.Type.EmptyTypes` to obtain a property that is not indexed.
modifiers	The only defined value for this parameter is `null`.

Return Value

A `System.Reflection.PropertyInfo` object reflecting the property that is defined in the type represented by the current instance and matches the specified criteria. If no matching property is found, returns `null`. If the type reflected by the current instance is contained in a loaded assembly, the property that matches the specified criteria is not public, and the caller does not have sufficient permission, returns `null`.

Description

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.
- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.
- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.
- `System.Reflection.BindingFlags.IgnoreCase` to ignore the case of *name*.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

This version of `System.Type.GetProperty` is equivalent to `System.Type.GetPropertyImpl`(*name*, *bindingAttr*, *binder*, *returnType*, *types*, *modifiers*).

Different programming languages use different syntax to specify indexed properties. Internally, this property is referred to by the name "Item" in the metadata. Therefore, any attempt to retrieve an indexed property using reflection is required to specify this internal name in order for the `PropertyInfo` to be returned correctly.

If the current instance represents a generic type, this method returns the `System.Reflection.PropertyInfo` with the type parameters replaced by the appropriate type arguments.

If the current instance represents an unassigned type parameter of a generic type or method, this method searches the properties of the class constraint; the properties of all interface constraints; and the properties of any interfaces inherited from class or interface constraints.

Exceptions

Exception	Condition
System.Reflection.AmbiguousMatchException	More than one property matching the specified criteria was found.
System.ArgumentNullException	*name* or *types* is `null`, or at least one of the elements in *types* is `null`.
System.ArgumentException	*types* has more than one dimension.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetPropertyImpl(System.String, System.Reflection.BindingFlags, System.Reflection.Binder, System.Type, System.Type[], System.Reflection.ParameterModifier[]) Method

```
[ILAsm]
.method family hidebysig virtual abstract class
System.Reflection.PropertyInfo GetPropertyImpl(string name, valuetype
System.Reflection.BindingFlags bindingAttr, class System.Reflection.Binder
binder, class System.Type returnType, class System.Type[] types, class
System.Reflection.ParameterModifier[] modifiers)

[C#]
protected abstract PropertyInfo GetPropertyImpl(string name, BindingFlags
bindingAttr, Binder binder, Type returnType, Type[] types,
ParameterModifier[] modifiers)
```

Summary

When overridden in a derived class implements the `System.Type.GetProperty` method and returns a `System.Reflection.PropertyInfo` object that reflects the property defined in the type represented by the current instance that matches the specified search criteria.

Parameters

Parameter	Description
name	A `System.String` containing the name of the property to be returned.
bindingAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, this method returns `null`.
binder	A `System.Reflection.Binder` object that defines a set of properties and enables the binding, coercion of argument types, and invocation of members using reflection. Specify `null` to use the `System.Type.DefaultBinder`.
returnType	A `System.Type` object that represents the type of the property to be returned.
types	An array of `System.Type` objects. The elements in the array are of the same

	number, in the same order, and represent the same types as the parameters for the indexer to be returned. Specify `System.Type.EmptyTypes` to obtain a property that is not indexed.
modifiers	The only defined value for this parameter is `null`.

Return Value

A `System.Reflection.PropertyInfo` object representing the property that matches the specified search criteria, if found; otherwise, `null`. If the type reflected by the current instance is from a loaded assembly, the matching property is not public, and the caller does not have permission to reflect on non-public objects in loaded assemblies, returns `null`.

Description

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.
- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.
- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.
- `System.Reflection.BindingFlags.IgnoreCase` to ignore the case of *name*.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

Behaviors

Different programming languages use different syntax to specify indexed properties. Internally, this property is referred to by the name "Item" in the metadata. Therefore, any attempt to retrieve an indexed property using reflection is required to specify this internal name in order for the `PropertyInfo` to be returned correctly.

Exceptions

Exception	Condition
System.Reflection.AmbiguousMatchException	More than one property matching the specified criteria was found.
System.ArgumentNullException	*name* or *types* is `null`, or at least one of the elements in *types* is `null`.
System.ArgumentException	*types* has more than one dimension.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetType(System.String, System.Boolean, System.Boolean) Method

```
[ILAsm]
.method public hidebysig static class System.Type GetType(string typeName,
bool throwOnError, bool ignoreCase)

[C#]
public static Type GetType(string typeName, bool throwOnError, bool
ignoreCase)
```

Summary

Returns the `System.Type` with the specified name, optionally performing a case-insensitive search and optionally throwing an exception if an error occurs while loading the `System.Type`.

Parameters

Parameter	Description
typeName	A `System.String` containing the name of the `System.Type` to return.
throwOnError	A `System.Boolean`. Specify `true` to throw a `System.TypeLoadException` if an error occurs while loading the `System.Type`. Specify `false` to ignore errors while loading the `System.Type`.
ignoreCase	A `System.Boolean`. Specify `true` to perform a case-insensitive search for *typeName*. Specify `false` to perform a case-sensitive search for *typeName*.

Return Value

The `System.Type` with the specified name, if found; otherwise, `null`. If the requested type is non-public and the caller does not have permission to reflect non-public objects outside the current assembly, this method returns `null`.

Description

typeName can be a simple type name, a fully qualified name, or a complex name that includes an assembly name. [*Note:* `System.Type.AssemblyQualifiedName` returns a fully qualified type name including nested types, the assembly name, and generic type arguments.]

If *typeName* includes only the name of the `System.Type`, this method searches in the calling object's assembly, then in the mscorlib.dll assembly. If *typeName* is fully qualified with the partial or complete assembly name, this method searches in the specified assembly.

[*Note:*

The following table shows calls to `GetType` for various types. (Some long strings have been wrapped to fit in the right column.)

To Get this Type	Use this String
An unmanaged pointer to `MyType`	`Type.GetType("MyType*")`
An unmanaged pointer to a pointer to `MyType`	`Type.GetType("MyType**")`
A managed pointer or reference to `MyType`	`Type.GetType("MyType&")`Note that unlike pointers, references are limited to one level.
A parent class and a nested class	`Type.GetType("MyParentClass+MyNestedClass")`
A one-dimensional array with a lower bound of 0	`Type.GetType("MyArray[]")`
A one-dimensional array with an unknown lower	`Type.GetType("MyArray[*]")`

bound	
An n-dimensional array	A comma (,) inside the brackets a total of n-1 times. For example, `System.Object[,,]` represents a three-dimensional `Object` array.
A two-dimensional array's array	`Type.GetType("MyArray[][]")`
A rectangular two-dimensional array with unknown lower bounds	`Type.GetType("MyArray[*,*]")` or `Type.GetType("MyArray[,]")`
A generic type with one type argument	`Type.GetType("MyGenericType[MyType]")`
A generic type with two type arguments	`Type.GetType("MyGenericType[MyType,AnotherType]")`
A generic type with two assembly-qualified type arguments	`Type.GetType("MyGenericType[[MyType,MyAssembly], [AnotherType,AnotherAssembly]]")`
An assembly-qualified generic type with an assembly-	`Type.GetType("MyGenericType[[MyType,MyAssembly]],MyGenericTypeAssembly")`

qualified type argument	
A generic type whose type argument is a generic type with two type arguments	`Type.GetType("MyGenericType[AnotherGenericType [MyType,AnotherType]]")`

]

Exceptions

Exception	Condition
System.ArgumentNullException	*typeName* is `null`.
System.Reflection.TargetInvocationException	A type initializer was invoked and threw an exception.
System.TypeLoadException	*throwOnError* is `true` and an error was encountered while loading the selected `System.Type`.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetType(System.String, System.Boolean) Method

```
[ILAsm]
.method public hidebysig static class System.Type GetType(string typeName,
bool throwOnError)

[C#]
public static Type GetType(string typeName, bool throwOnError)
```

Summary

Returns the `System.Type` with the specified name, optionally throwing an exception if an error occurs while loading the `System.Type`.

Parameters

Parameter	Description
typeName	A `System.String` containing the case-sensitive name of the `System.Type` to return.
throwOnError	A `System.Boolean`. Specify `true` to throw a `System.TypeLoadException` if an error occurs while loading the `System.Type`. Specify `false` to ignore errors while loading the `System.Type`.

Return Value

The `System.Type` with the specified name, if found; otherwise, `null`. If the requested type is non-public and the caller does not have permission to reflect non-public objects outside the current assembly, this method returns `null`.

Description

This method is equivalent to `System.Type.GetType`(*name*, *throwOnError*, `false`).

typeName can be a simple type name, a fully qualified name, or a complex name that includes an assembly name specification. If *typeName* includes only the name of the `System.Type`, this method searches in the calling object's assembly, then in the mscorlib.dll assembly. If *typeName* is fully qualified with the partial or complete assembly name, this method searches in the specified assembly.

[*Note:* `System.Type.AssemblyQualifiedName` can return a fully qualified type name including nested types, the assembly name, and generic type arguments. For complete details, see `System.Type.GetType`(`System.String`, `System.Boolean`,

System.Boolean).]

Exceptions

Exception	Condition
System.ArgumentNullException	*typeName* is `null`.
System.Reflection.TargetInvocationException	A type initializer was invoked and threw an exception.
System.TypeLoadException	*throwOnError* is `true` and an error was encountered while loading the `System.Type`.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public objects. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetType(System.String) Method

```
[ILAsm]
.method public hidebysig static class System.Type GetType(string typeName)

[C#]
public static Type GetType(string typeName)
```

Summary

Returns the `System.Type` with the specified name.

Parameters

Parameter	Description
typeName	A `System.String` containing the case-sensitive name of the `System.Type` to return.

Return Value

The `System.Type` with the specified name, if found; otherwise, `null`. If the requested type is non-public and the caller does not have permission to reflect non-public objects outside the current assembly, this method returns `null`.

Description

This method is equivalent to `System.Type.GetType`(*name*, `false`, `false`).

typeName can be a simple type name, a type name that includes a namespace, or a complex name that includes an assembly name specification. If *typeName* includes only the name of the `System.Type`, this method searches in the calling object's assembly, then in the mscorlib.dll assembly. If *typeName* is fully qualified with the partial or complete assembly name, this method searches in the specified assembly.

[*Note:* `System.Type.AssemblyQualifiedName` can return a fully qualified type name including nested types, the assembly name, and generic type arguments. For complete details, see `System.Type.GetType`(`System.String`, `System.Boolean`, `System.Boolean`).]

Exceptions

Exception	Condition

System.ArgumentNullException	*typeName* is `null`.
System.Reflection. TargetInvocationException	A type initializer was invoked and threw an exception.

1

2 **Permissions**

Permission	Description
System.Security.Permissions. ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

3

4

The following member must be implemented if the Reflection library is present in the implementation.

Type.GetTypeArray(System.Object[]) Method

```
[ILAsm]
.method public hidebysig static class System.Type[] GetTypeArray(object[]
args)

[C#]
public static Type[] GetTypeArray(object[] args)
```

Summary

Returns the types of the objects in the specified array.

Parameters

Parameter	Description
args	An array of objects whose types are to be returned.

Return Value

An array of `System.Type` objects representing the types of the corresponding elements in *args*. If a requested type is not public and the caller does not have permission to reflect non-public objects outside the current assembly, the corresponding element in the array returned by this method will be `null`.

Exceptions

Exception	Condition
System.ArgumentNullException	*args* is `null`.
System.Reflection.TargetInvocationException	The type initializers were invoked and at least one threw an exception.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions`.

	ReflectionPermissionFlag.TypeInformation.

1

2

The following member must be implemented if the RuntimeInfrastructure library is present in the implementation.

Type.GetTypeFromHandle(System.RuntimeTypeHandle) Method

```
[ILAsm]
.method public hidebysig static class System.Type
GetTypeFromHandle(valuetype System.RuntimeTypeHandle handle)

[C#]
public static Type GetTypeFromHandle(RuntimeTypeHandle handle)
```

Summary

Gets the `System.Type` referenced by the specified type handle.

Parameters

Parameter	Description
handle	The `System.RuntimeTypeHandle` object that refers to the desired `System.Type`.

Return Value

The `System.Type` referenced by the specified `System.RuntimeTypeHandle`.

Description

The handles are valid only in the application domain in which they were obtained.

Exceptions

Exception	Condition
System.ArgumentNullException	*handle* is `null`.
System.Security.SecurityException	The requested type is non-public and outside the current assembly, and the caller does not have the required permission.
System.Reflection.TargetInvocationException	A type initializer was invoked and threw an exception.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public objects. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`

1

2

The following member must be implemented if the RuntimeInfrastructure library is present in the implementation.

Type.GetTypeHandle(System.Object) Method

```
[ILAsm]
.method public hidebysig static valuetype System.RuntimeTypeHandle
GetTypeHandle(object o)

[C#]
public static RuntimeTypeHandle GetTypeHandle(object o)
```

Summary

Returns the handle for the `System.Type` of the specified object.

Parameters

Parameter	Description
o	The object for which to get the type handle.

Return Value

The `System.RuntimeTypeHandle` for the `System.Type` of the specified `System.Object`.

Description

The handle is valid only in the application domain in which it was obtained.

The following member must be implemented if the Reflection library is present in the implementation.

Type.HasElementTypeImpl() Method

```
[ILAsm]
.method family hidebysig virtual abstract bool HasElementTypeImpl()

[C#]
protected abstract bool HasElementTypeImpl()
```

Summary

When overridden in a derived class, implements the `System.Type.HasElementType` property and determines whether the current `System.Type` encompasses or refers to another type; that is, whether the current `System.Type` is an array, a pointer, or is passed by reference.

Return Value

`true` if the `System.Type` is an array, a pointer, or is passed by reference; otherwise, `false`.

Description

[*Note:* For example, `System.Type.GetType` ("System.Int32[]").HasElementTypeImpl returns `true`, but `System.Type.GetType` ("System.Int32").HasElementTypeImpl returns `false`. `System.Type.HasElementTypeImpl` also returns `true` for "System.Int32*" and "System.Int32&".]

The following member must be implemented if the Reflection library is present in the implementation.

Type.InvokeMember(System.String, System.Reflection.BindingFlags, System.Reflection.Binder, System.Object, System.Object[], System.Globalization.CultureInfo) Method

```
[ILAsm]
.method public hidebysig instance object InvokeMember(string name,
valuetype System.Reflection.BindingFlags invokeAttr, class
System.Reflection.Binder binder, object target, object[] args, class
System.Globalization.CultureInfo culture)

[C#]
public object InvokeMember(string name, BindingFlags invokeAttr, Binder
binder, object target, object[] args, CultureInfo culture)
```

Summary

Invokes the specified member, using the specified binding constraints and matching the specified argument list and culture.

Parameters

Parameter	Description
name	A `System.String` containing the name of the constructor or method to invoke, or property or field to access. If the type represented by the current instance has a default member, specify `System.String.Empty` to invoke that member. [*Note:* For more information on default members, see `System.Reflection.DefaultMemberAttribute`.]
invokeAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, `System.Reflection.BindingFlags.Public` | `System.Reflection.BindingFlags.Instance` is used by default.
binder	A `System.Reflection.Binder` object that defines a set of properties and enables the binding, coercion of argument types, and invocation of members using reflection. Specify `null` to use the `System.Type.DefaultBinder`.

target	A `System.Object` on which to invoke the member that matches the other specified criteria. If the matching member is `static`, this parameter is ignored.
args	An array of objects containing the arguments to pass to the member to be invoked. The elements of this array are of the same number and in the same order by assignment-compatible type as specified by the contract of the member to be bound. Specify an empty array or `null` for a member that has no parameters.
culture	The only defined value for this parameter is `null`.

Return Value

A `System.Object` containing the return value of the invoked member. If the invoked member does not have a return value, returns a `System.Object` containing `System.Void`.

Description

This version of `System.Type.InvokeMember` is equivalent to `System.Type.InvokeMember`(*name*, *invokeAttr*, *binder*, *target*, *args*, `null`, *culture*, `null`).

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.
System.ArgumentException	*args* has more than one dimension. -or- *invokeAttr* is not a valid `System.Reflection.BindingFlags` value. -or- The member to be invoked is a constructor and `System.Reflection.BindingFlags.CreateInstance` is not specified in *invokeAttr*. -or- The member to be invoked is a method that is not a

	type initializer or instance constructor, and `System.Reflection.BindingFlags.InvokeMethod` is not specified in *invokeAttr*. -or- The member to be accessed is a field, and neither `System.Reflection.BindingFlags.GetField` nor `System.Reflection.BindingFlags.SetField` is specified in *invokeAttr*. -or- The member to be accessed is a property, and neither `System.Reflection.BindingFlags.GetProperty` nor `System.Reflection.BindingFlags.SetProperty` is specified in *invokeAttr*. -or- *invokeAttr* contains `System.Reflection.BindingFlags.CreateInstance` and at least one of `System.Reflection.BindingFlags.InvokeMethod`, `System.Reflection.BindingFlags.GetField`, `System.Reflection.BindingFlags.SetField`, `System.Reflection.BindingFlags.GetProperty`, or `System.Reflection.BindingFlags.SetProperty`. -or- *invokeAttr* contains both `System.Reflection.BindingFlags.GetField` and `System.Reflection.BindingFlags.SetField`. -or- *invokeAttr* contains both `System.Reflection.BindingFlags.GetProperty` and `System.Reflection.BindingFlags.SetProperty`. -or- *invokeAttr* contains `System.Reflection.BindingFlags.InvokeMethod`and at least one of

	`System.Reflection.BindingFlags.SetField` or `System.Reflection.BindingFlags.SetProperty`. -or- *invokeAttr* contains `System.Reflection.BindingFlags.SetField` and *args* has more than one element.
System.MissingFieldException	A field or property matching the specified criteria was not found.
System.MissingMethodException	A method matching the specified criteria was not found. -or- The current instance object represents a type that contains open type parameters (that is, `System.Type.ContainsGenericParameters` returns `true`).
System.MethodAccessException	The requested member is non-public and the caller does not have the required permission.
System.Reflection.TargetException	The member matching the specified criteria cannot be invoked on *target*.
System.Reflection. TargetInvocationException	The member matching the specified criteria threw an exception.
System.Reflection. AmbiguousMatchException	More than one member matches the specified criteria.

Example

For an example that demonstrates `System.Type.InvokeMember`, see `System.Type.InvokeMember( System.String, System.Reflection.BindingFlags, System.Reflection.Binder, System.Object, System.Object[], System.Reflection.ParameterModifier[], System.Globalization.CultureInfo, System.String[])`.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

1

2

The following member must be implemented if the Reflection library is present in the implementation.

Type.InvokeMember(System.String, System.Reflection.BindingFlags, System.Reflection.Binder, System.Object, System.Object[]) Method

```
[ILAsm]
.method public hidebysig instance object InvokeMember(string name,
valuetype System.Reflection.BindingFlags invokeAttr, class
System.Reflection.Binder binder, object target, object[] args)

[C#]
public object InvokeMember(string name, BindingFlags invokeAttr, Binder
binder, object target, object[] args)
```

Summary

Invokes the specified member, using the specified binding constraints and matching the specified argument list.

Parameters

Parameter	Description
name	A `System.String` containing the name of the constructor or method to invoke, or property or field to access. If the type represented by the current instance has a default member, specify `System.String.Empty` to invoke that member. [*Note:* For more information on default members, see `System.Reflection.DefaultMemberAttribute`.]
invokeAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, `System.Reflection.BindingFlags.Public` | `System.Reflection.BindingFlags.Instance` is used by default.
binder	A `System.Reflection.Binder` object that defines a set of properties and enables the binding, coercion of argument types, and invocation of members using reflection. Specify `null` to use the `System.Type.DefaultBinder`.
target	A `System.Object` on which to invoke the member that matches the other specified criteria. If the matching member is `static`, this parameter is ignored.

args	An array of objects containing the arguments to pass to the member to be invoked. The elements of this array are of the same number and in the same order by assignment-compatible type as specified by the contract of the member to be bound. Specify an empty array or `null` for a member that has no parameters.

Return Value

A `System.Object` containing the return value of the invoked member. If the invoked member does not have a return value, returns a `System.Object` containing `System.Void`.

Description

This version of `System.Type.InvokeMember` is equivalent to `System.Type.InvokeMember`(*name*, *invokeAttr*, *binder*, *target*, *args*, `null`, `null`, `null`).

[*Note:* For a demonstration of the use of `System.Type.InvokeMember`, see the example for `System.Type.InvokeMember`(`System.String`, `System.Reflection.BindingFlags`, `System.Reflection.Binder`, `System.Object`, `System.Object`[], `System.Reflection.ParameterModifier`[], `System.Globalization.CultureInfo`, `System.String`[]).]

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.
System.ArgumentException	*args* has more than one dimension. -or- *invokeAttr* is not a valid `System.Reflection.BindingFlags` value. -or- The member to be invoked is a constructor and `System.Reflection.BindingFlags.CreateInstance` is not specified in *invokeAttr*. -or-

	The member to be invoked is a method that is not a type initializer or instance constructor, and `System.Reflection.BindingFlags.InvokeMethod` is not specified in *invokeAttr*. -or- The member to be accessed is a field, and neither `System.Reflection.BindingFlags.GetField` nor `System.Reflection.BindingFlags.SetField` is specified in *invokeAttr*. -or- The member to be accessed is a property, and neither `System.Reflection.BindingFlags.GetProperty` nor `System.Reflection.BindingFlags.SetProperty` is specified in *invokeAttr*. -or- *invokeAttr* contains `System.Reflection.BindingFlags.CreateInstance` and at least one of `System.Reflection.BindingFlags.InvokeMethod`, `System.Reflection.BindingFlags.GetField`, `System.Reflection.BindingFlags.SetField`, `System.Reflection.BindingFlags.GetProperty`, or `System.Reflection.BindingFlags.SetProperty`. -or- *invokeAttr* contains both `System.Reflection.BindingFlags.GetField` and `System.Reflection.BindingFlags.SetField`. -or- *invokeAttr* contains both `System.Reflection.BindingFlags.GetProperty` and `System.Reflection.BindingFlags.SetProperty`. -or-

	invokeAttr contains `System.Reflection.BindingFlags.InvokeMethod` and at least one of `System.Reflection.BindingFlags.SetField` or `System.Reflection.BindingFlags.SetProperty`. -or- *invokeAttr* contains `System.Reflection.BindingFlags.SetField` and *args* has more than one element.
System.MissingFieldException	A field or property matching the specified criteria was not found.
System.MissingMethodException	A method matching the specified criteria cannot be found. -or- The current instance object represents a type that contains open type parameters (that is, `System.Type.ContainsGenericParameters` returns `true`).
System.MethodAccessException	The requested member is non-public and the caller does not have the required permission.
System.Reflection.TargetException	The member matching the specified criteria cannot be invoked on *target*.
System.Reflection. TargetInvocationException	The member matching the specified criteria threw an exception.
System.Reflection. AmbiguousMatchException	More than one member matches the specified criteria.

Permissions

Permission	Description

System.Security.Permissions. ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions. ReflectionPermissionFlag.TypeInformation`.

1

2

The following member must be implemented if the Reflection library is present in the implementation.

Type.InvokeMember(System.String, System.Reflection.BindingFlags, System.Reflection.Binder, System.Object, System.Object[], System.Reflection.ParameterModifier[], System.Globalization.CultureInfo, System.String[]) Method

```
[ILAsm]
.method public hidebysig virtual abstract object InvokeMember(string name,
valuetype System.Reflection.BindingFlags invokeAttr, class
System.Reflection.Binder binder, object target, object[] args, class
System.Reflection.ParameterModifier[] modifiers, class
System.Globalization.CultureInfo culture, string[] namedParameters)

[C#]
public abstract object InvokeMember(string name, BindingFlags invokeAttr,
Binder binder, object target, object[] args, ParameterModifier[]
modifiers, CultureInfo culture, string[] namedParameters)
```

Summary

Invokes or accesses a member defined on the type represented by the current instance that matches the specified binding criteria.

Parameters

Parameter	Description
name	A `System.String` containing the name of the constructor or method to invoke, or property or field to access. If the type represented by the current instance has a default member, specify `System.String.Empty` to invoke that member. [*Note:* For more information on default members, see `System.Reflection.DefaultMemberAttribute`.]
invokeAttr	A bitwise combination of `System.Reflection.BindingFlags` values that control the binding process. If zero is specified, `System.Reflection.BindingFlags.Public` | `System.Reflection.BindingFlags.Instance` is used by default.

binder	A `System.Reflection.Binder` that defines a set of properties, and enables the binding, coercion of argument types, and invocation of members using reflection. Specify `null` to use`System.Type.DefaultBinder`.
target	A `System.Object` on which to invoke the member that matches the other specified criteria. If the matching member is `static`, this parameter is ignored.
args	An array of objects containing the arguments to pass to the member to be invoked. The elements of this array are of the same number and in the same order by assignment-compatible type as specified by the contract of the member to be bound if and only if *nameParameters* is `null`. If *namedParameters* is not `null`, the order of the elements in *args* corresponds to the order of the parameters specified in *namedParameters*. Specify an empty array or `null` for a member that takes no parameters.
modifiers	The only defined value for this parameter is `null`.
culture	The only defined value for this parameter is `null`.
namedParameters	An array of `System.String` objects containing the names of the parameters to which the values in *args* are passed. These names are processed in a case-sensitive manner and have a one-to-one correspondence with the elements of *args*. Specify an empty array or `null` for a member that takes no parameters. Specify `null` to have this parameter ignored.

Return Value

A `System.Object` containing the return value of the invoked or accessed member. If the member does not have a return value, returns a `System.Object` containing `System.Void`.

Description

`System.Type.InvokeMember` calls a constructor or a method, gets or sets a property, gets or sets a field, or gets or sets an element of an array.

The binder finds all of the matching members. These members are found based upon the type of binding specified by *invokeAttr*. The `System.Reflection.Binder.BindToMethod` is responsible for selecting the method to be invoked. The default binder selects the most specific match. The set of members is then filtered by name, number of arguments, and a set of search modifiers defined in the binder. After the member is selected, it is invoked or accessed. Accessibility is

checked at that point. Access restrictions are ignored for fully trusted code; that is, private constructors, methods, fields, and properties can be accessed and invoked via reflection whenever the code is fully trusted.

The following `System.Reflection.BindingFlags` are used to define which members to include in the search:

- Specify either `System.Reflection.BindingFlags.Instance` or `System.Reflection.BindingFlags.Static` to get a return value other than `null`.
- Specify `System.Reflection.BindingFlags.Public` to include public members in the search.
- Specify `System.Reflection.BindingFlags.NonPublic` to include non-public members (that is, private and protected members) in the search.

The following `System.Reflection.BindingFlags` values can be used to change how the search works:

- `System.Reflection.BindingFlags.DeclaredOnly` to search only the members declared in the type, not members that were simply inherited.
- `System.Reflection.BindingFlags.IgnoreCase` to ignore the case of *name*.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

Behaviors

Each parameter in the *namedParameters* array is assigned the value in the corresponding element in the *args* array. If the length of *args* is greater than the length of *namedParameters*, the remaining argument values are passed in order.

A member will be found only if the number of parameters in the member declaration equals the number of arguments in the args array (unless default arguments are defined on the member). Also, The type of each argument is required to be convertible by the binder to the type of the parameter.

It is required that the caller specify values for *invokeAttr* as follows:

Action	BindingFlags
Invoke a constructor.	`System.Reflection.BindingFlags.CreateInstance`. This flag is not valid with the other flags in this table. If this flag is specified, *name* is ignored.
Invoke a method.	`System.Reflection.BindingFlags.InvokeMethod`. This flag if not valid with `System.Reflection.BindingFlags.CreateInstance`, `System.Reflection.BindingFlags.SetField`, or

	`System.Reflection.BindingFlags.SetProperty`.
Define a field value.	`System.Reflection.BindingFlags.SetField`. This flag is not valid with `System.Reflection.BindingFlags.CreateInstance`, `System.Reflection.BindingFlags.InvokeMethod`, or `System.Reflection.BindingFlags.GetField`.
Return a field value.	`System.Reflection.BindingFlags.GetField`. This flag is not valid with `System.Reflection.BindingFlags.CreateInstance`, `System.Reflection.BindingFlags.InvokeMethod`, or `System.Reflection.BindingFlags.SetField`.
Set a property.	`System.Reflection.BindingFlags.SetProperty`. This flag is not valid with `System.Reflection.BindingFlags.CreateInstance`, `System.Reflection.BindingFlags.InvokeMethod`, or `System.Reflection.BindingFlags.GetProperty`.
Get a property.	`System.Reflection.BindingFlags.GetProperty`. This flag is not valid with `System.Reflection.BindingFlags.CreateInstance`, `System.Reflection.BindingFlags.InvokeMethod`, or `System.Reflection.BindingFlags.SetProperty`.

[*Note:* For more information, see `System.Reflection.BindingFlags`.]

Usage

`System.Type.InvokeMember` can be used to invoke methods with parameters that have default values. To bind to these methods, `System.Reflection.BindingFlags.OptionalParamBinding` must be specified. For a parameter that has a default value, the caller can supply a value or supply `System.Type.Missing` to use the default value.

`System.Type.InvokeMember` can be used to set a field to a particular value by specifying `System.Reflection.BindingFlags.SetField`. For example, to set a public instance field named F on class C, where F is a string, the value is set using the following statement:

```
typeof(C).InvokeMember("F", BindingFlags.SetField, null, C, new Object{
"strings new value"}, null, null, null);
```

A string array F can be initialized as follows:

```
typeof(C).InvokeMember("F", BindingFlags.SetField, null, C, new Object{new
```

```
String[]{"a","z","c","d"}, null, null, null);


```

Use `System.Type.InvokeMember` to set the value of an element in an array by specifying the index of the value and the new value for the element as follows:

```
typeof(C).InvokeMember("F", BindingFlags.SetField, null, C, new Object{1,
"b"}, null, null, null);

```

The preceding statement changes "z" in array F to "b".

Exceptions

Exception	Condition
System.ArgumentNullException	*name* is `null`.
System.ArgumentException	*args* has more than one dimension. -or- *invokeAttr* is not a valid `System.Reflection.BindingFlags` value. -or- The member to be invoked is a constructor and `System.Reflection.BindingFlags.CreateInstance` is not specified in *invokeAttr*. -or- The member to be invoked is a method that is not a type initializer or instance constructor, and `System.Reflection.BindingFlags.InvokeMethod` is not specified in *invokeAttr*. -or- The member to be accessed is a field, and neither `System.Reflection.BindingFlags.GetField` nor `System.Reflection.BindingFlags.SetField` is specified in *invokeAttr*. -or- The member to be accessed is a property, and

	neither `System.Reflection.BindingFlags.GetProperty` nor `System.Reflection.BindingFlags.SetProperty` is specified in *invokeAttr*. -or- *invokeAttr* contains `System.Reflection.BindingFlags.CreateInstance` and at least one of `System.Reflection.BindingFlags.InvokeMethod`, `System.Reflection.BindingFlags.GetField`, `System.Reflection.BindingFlags.SetField`, `System.Reflection.BindingFlags.GetProperty`, or `System.Reflection.BindingFlags.SetProperty`. -or- *invokeAttr* contains both `System.Reflection.BindingFlags.GetField` and `System.Reflection.BindingFlags.SetField`. -or- *invokeAttr* contains both `System.Reflection.BindingFlags.GetProperty` and `System.Reflection.BindingFlags.SetProperty`. -or- *invokeAttr* contains `System.Reflection.BindingFlags.InvokeMethod` and at least one of `System.Reflection.BindingFlags.SetField` or `System.Reflection.BindingFlags.SetProperty`. -or- *invokeAttr* contains `System.Reflection.BindingFlags.SetField` and *args* has more than one element. -or-

	namedParameters.Length > *args*.Length. -or- At least one element in *namedParameters* is `null`. -or- At least one element in *args* is not assignment-compatible with the corresponding parameter in *namedParameters*.
System.MissingFieldException	A field or property matching the specified criteria was not found.
System.MissingMethodException	A method matching the specified criteria cannot be found. -or- The current instance object represents a type that contains open type parameters (that is, `System.Type.ContainsGenericParameters` returns `true`).
System.MethodAccessException	The requested member is non-public and the caller does not have the required permission.
System.Reflection.TargetException	The member matching the specified criteria cannot be invoked on *target*.
System.Reflection. TargetInvocationException	The member matching the specified criteria threw an exception.
System.Reflection. AmbiguousMatchException	More than one member matches the specified criteria.

Example

The following example demonstrates the use of `System.Type.InvokeMember` to construct a `System.String`, obtain its `System.String.Length` property, invoke `System.String.Insert` on it, and then set its value using the `System.String.Empty` field.

[C#]

```
using System;
using System.Reflection;

class InvokeMemberExample
{
   static void Main(string[] args)
   {
      // Create the parameter arrays that will
      // be passed to InvokeMember.
      char[] cAry =
      new char[] {'A',' ','s','t','r','i','n','g'};
      object[] oAry = new object[] {cAry, 0, cAry.Length};

      Type t = typeof(string);

      // Invoke the constructor of a string.
      string str =
         (string)t.InvokeMember(null, BindingFlags.Instance |
         BindingFlags.Public | BindingFlags.CreateInstance, null,
         null, oAry, null, null, null);
      Console.WriteLine("The string is \"{0}\".", str);

      // Access a property of the string.
      int i =
         (int) t.InvokeMember("Length", BindingFlags.Instance |
         BindingFlags.Public | BindingFlags.GetProperty, null,
         str, null, null, null, null);
      Console.WriteLine("The length of the string is {0}.", i);

      // Invoke a method on the string.
      string newStr = "new ";
      object[] oAry2 = new Object[] {2, newStr};
      str = (string) t.InvokeMember("Insert", BindingFlags.Instance |
         BindingFlags.Public | BindingFlags.InvokeMethod, null, str,
         oAry2, null, null, null);
      Console.WriteLine("The modified string is \"{0}\".", str);

      // Access a field of the string.
      str = (string) t.InvokeMember("Empty", BindingFlags.Static |
         BindingFlags.Public | BindingFlags.GetField, null, str,
         null);
      Console.WriteLine("The empty string is \"{0}\".", str);

   }
}
```

The output is

```
The string is "A string".


The length of the string is 8.
```

```
The modified string is "A new string"


The empty string is "".
```

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to retrieve information on non-public members of types in loaded assemblies. See `System.Security.Permissions.ReflectionPermissionFlag.TypeInformation`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsArrayImpl() Method

```
[ILAsm]
.method family hidebysig virtual abstract bool IsArrayImpl()

[C#]
protected abstract bool IsArrayImpl()
```

Summary

When overridden in a derived class implements the `System.Type.IsArray` property returning a `System.Boolean` value that indicates whether the type represented by the current instance is an array.

Return Value

`true` if the `System.Type` is an array; otherwise, `false`.

Description

An instance of the `System.Array` class is required to return `false` because it is an object, not an array.

Behaviors

As described above.

Type.IsAssignableFrom(System.Type) Method

```
[ILAsm]
.method public hidebysig virtual bool IsAssignableFrom(class System.Type
c)

[C#]
public virtual bool IsAssignableFrom(Type c)
```

Summary

Determines whether an instance of the current `System.Type` can be assigned from an instance of the specified `System.Type`.

Parameters

Parameter	Description
c	The `System.Type` to compare with the current `System.Type`.

Return Value

`false` if *c* is a null reference.

`true` if one or more of the following statements are true; otherwise `false`.

- If *c* and the current `System.Type` represent the same type.
- If the current `System.Type` is in the inheritance hierarchy of *c*.
- If the current `System.Type` is an interface and *c* supports that interface.
- If *c* is a generic type parameter and the current instance represents one of the constraints of *c*.

Description

[*Note:* A generic type definition is not assignable from a closed constructed type.

]

Example

The following example demonstrates the `System.Type.IsAssignableFrom` method using arrays.

[C#]

```
using System;
class ArrayTypeTest {
 public static void Main() {
 int i = 1;
 int [] array10 = new int [10];
 int [] array2 = new int[2];
 int [,]array22 = new int[2,2];
 int [,]array24 = new int[2,4];
 int [,,]array333 = new int[3,3,3];
 Type array10Type = array10.GetType();
 Type array2Type = array2.GetType();
 Type array22Type = array22.GetType();
 Type array24Type = array24.GetType();
 Type array333Type = array333.GetType();

 // If X and Y are not both arrays, then false
 Console.WriteLine("int[2] is assignable from int? {0} ",
array2Type.IsAssignableFrom(i.GetType()));
 // If X and Y have same type and rank, then true.
 Console.WriteLine("int[2] is assignable from int[10]? {0} ",
array2Type.IsAssignableFrom(array10Type));
 Console.WriteLine("int[2,2] is assignable from int[2,4]? {0}",
array22Type.IsAssignableFrom(array24Type));
 Console.WriteLine("int[2,4] is assignable from int[2,2]? {0}",
array24Type.IsAssignableFrom(array22Type));
 Console.WriteLine("");
 // If X and Y do not have the same rank, then false.
 Console.WriteLine("int[2,2] is assignable from int[10]? {0}",
array22Type.IsAssignableFrom(array10Type));
 Console.WriteLine("int[2,2] is assignable from int[3,3,3]? {0}",
array22Type.IsAssignableFrom(array333Type));
 Console.WriteLine("int[3,3,3] is assignable from int[2,2]? {0}",
array333Type.IsAssignableFrom(array22Type));
 }
}
```

The output is

```
int[2] is assignable from int? False


int[2] is assignable from int[10]? True


int[2,2] is assignable from int[2,4]? True


int[2,4] is assignable from int[2,2]? True


int[2,2] is assignable from int[10]? False

```

```
1
2   int[2,2] is assignable from int[3,3,3]? False
3
4
5   int[3,3,3] is assignable from int[2,2]? False
6

7
```

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsByRefImpl() Method

```
[ILAsm]
.method family hidebysig virtual abstract bool IsByRefImpl()

[C#]
protected abstract bool IsByRefImpl()
```

Summary

When overridden in a derived class, implements the `System.Type.IsByRef` property and determines whether the `System.Type` is passed by reference.

Return Value

`true` if the `System.Type` is passed by reference; otherwise, `false`.

Behaviors

As described above.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsCOMObjectImpl() Method

```
[ILAsm]
.method family hidebysig virtual abstract bool IsCOMObjectImpl()

[C#]
protected abstract bool IsCOMObjectImpl()
```

Summary

Reserved.

Return Value

`false`

Description

This abstract method is required to be present for legacy implementations. Conforming implementations are permitted to throw the `System.NotSupportedException` as their implementation.

Type.IsInstanceOfType(System.Object) Method

```
[ILAsm]
.method public hidebysig virtual bool IsInstanceOfType(object o)

[C#]
public virtual bool IsInstanceOfType(object o)
```

Summary

Determines whether the specified object is an instance of the current `System.Type`.

Parameters

Parameter	Description
o	The object to compare with the current `System.Type`.

Return Value

`true` if either of the following statements is true; otherwise `false`.

- If the current `System.Type` is in the inheritance hierarchy of *o*.
- If the current `System.Type` is an interface and *o* supports that interface.

If *o* is a null reference or if the current instance is an open generic type (that is, `System.Type.ContainsGenericParameters` returns `true`) returns `false`.

Description

As described above.

[*Note:* A constructed type is not an instance of its generic type definition.

]

Behaviors

As described above.

Example

The following example demonstrates the `System.Type.IsInstanceOfType` method.

[C#]

```
using System;
public interface IFoo { }
public class MyClass: IFoo {}
public class MyDerivedClass: MyClass {}
class IsInstanceTest {
 public static void Main() {
 Type ifooType=typeof(IFoo);
 MyClass mc = new MyClass();
 Type mcType = mc.GetType();
 MyClass mdc = new MyDerivedClass();
 Type mdcType = mdc.GetType();
 int [] array = new int [10];
 Type arrayType = typeof(Array);
 Console.WriteLine("int[] is instance of Array? {0}",
arrayType.IsInstanceOfType(array));
 Console.WriteLine("myclass instance is instance of MyClass? {0}",
mcType.IsInstanceOfType(mc));
 Console.WriteLine("myderivedclass instance is instance of MyClass? {0}",
mcType.IsInstanceOfType(mdc));
 Console.WriteLine("myclass instance is instance of IFoo? {0}",
ifooType.IsInstanceOfType(mc));
 Console.WriteLine("myderivedclass instance is instance of IFoo? {0}",
ifooType.IsInstanceOfType(mdc));
 }
}
```

The output is

```
int[] is instance of Array? True


myclass instance is instance of MyClass? True


myderivedclass instance is instance of MyClass? True


myclass instance is instance of IFoo? True


myderivedclass instance is instance of IFoo? True
```

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsPointerImpl() Method

```
[ILAsm]
.method family hidebysig virtual abstract bool IsPointerImpl()

[C#]
protected abstract bool IsPointerImpl()
```

Summary

When overridden in a derived class, implements the `System.Type.IsPointer` property and determines whether the `System.Type` is a pointer.

Return Value

`true` if the `System.Type` is a pointer; otherwise, `false`.

Behaviors

As described above.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsPrimitiveImpl() Method

```
[ILAsm]
.method family hidebysig virtual abstract bool IsPrimitiveImpl()

[C#]
protected abstract bool IsPrimitiveImpl()
```

Summary

When overridden in a derived class, implements the `System.Type.IsPrimitive` property and determines whether the `System.Type` is one of the primitive types.

Return Value

`true` if the `System.Type` is one of the primitive types; otherwise, `false`.

Behaviors

This method returns `true` if the underlying type of the current instance is one of the following: `System.Boolean`, `System.Byte`, `System.SByte`, `System.Int16`, `System.UInt16`, `System.Int32`, `System.UInt32`, `System.Int64`, `System.UInt64`, `System.Char`, `System.Double`, and `System.Single`.

Type.IsSubclassOf(System.Type) Method

```
[ILAsm]
.method public hidebysig virtual bool IsSubclassOf(class System.Type c)

[C#]
public virtual bool IsSubclassOf(Type c)
```

Summary

Determines whether the current `System.Type` derives from the specified `System.Type`.

Parameters

Parameter	Description
c	The `System.Type` to compare with the current `System.Type`.

Return Value

`true` if *c* and the current `System.Type` represent classes, and the class represented by the current `System.Type` derives from the class represented by *c*; otherwise `false`. Returns `false` if *c* and the current `System.Type` represent the same class.

Description

Interfaces are not considered.

If the current instance represents an unassigned type parameter of a generic type or method, it derives from its class constraint, or from `System.Object` if it has no class constraint.

Example

The following example demonstrates the `System.Type.IsSubclassOf` method.

[C#]

```
using System;
public interface IFoo { }
public interface IBar:IFoo{}
public class MyClass: IFoo {}
public class MyDerivedClass: MyClass {}
class IsSubclassTest {
 public static void Main() {
 Type ifooType = typeof(IFoo);
 Type ibarType = typeof(IBar);
 MyClass mc = new MyClass();
 Type mcType = mc.GetType();
```

```
 1    MyClass mdc = new MyDerivedClass();
 2    Type mdcType = mdc.GetType();
 3    int [] array = new int [10];
 4    Type arrayOfIntsType = array.GetType();
 5    Type arrayType = typeof(Array);
 6
 7    Console.WriteLine("Array is subclass of int[]? {0}",
 8   arrayType.IsSubclassOf(arrayOfIntsType));
 9    Console.WriteLine("int [] is subclass of Array? {0}",
10   arrayOfIntsType.IsSubclassOf(arrayType));
11    Console.WriteLine("IFoo is subclass of IBar? {0}",
12   ifooType.IsSubclassOf(ibarType));
13    Console.WriteLine("myclass is subclass of MyClass? {0}",
14   mcType.IsSubclassOf(mcType));
15    Console.WriteLine("myderivedclass is subclass of MyClass? {0}",
16   mdcType.IsSubclassOf(mcType));
17    Console.WriteLine("IBar is subclass of IFoo? {0}",
18   ibarType.IsSubclassOf(ifooType));
19     }
20   }
21   The output is
22
23   Array is subclass of int[]? False
24
25
26   int [] is subclass of Array? True
27
28
29   IFoo is subclass of IBar? False
30
31
32   myclass is subclass of MyClass? False
33
34
35   myderivedclass is subclass of MyClass? True
36
37
38   IBar is subclass of IFoo? False
39


40
```

The following member must be implemented if the Reflection library is present in the implementation.

Type.MakeArrayType() Method

```
[ILAsm]
.method public hidebysig virtual instance class System.Type
MakeArrayType()

[C#]
public virtual Type MakeArrayType()
```

Summary

Returns a `System.Type` object representing a one-dimensional array type whose element type is the current type, with a lower bound of zero.

Return Value

A `System.Type` object representing a one-dimensional array type whose element type is the current type, with a lower bound of zero.

Description

This method provides a way to generate an array type with any possible element type, including generic types.

The following member must be implemented if the Reflection library is present in the implementation.

Type.MakeArrayType(System.Int32) Method

```
[ILAsm]
.method public hidebysig virtual instance class System.Type
MakeArrayType(int32 rank)

[C#]
public virtual Type MakeArrayType(int rank)
```

Summary

Returns a `System.Type` object representing an array of the current type, with the specified number of dimensions.

Parameters

Parameter	Description
rank	The number of dimensions for the array.

Return Value

A `System.Type` object representing an array of the current type, with the specified number of dimensions.

Description

This method provides a way to generate an array with any possible element type, including generic types.

Exceptions

Exception	Condition
System.IndexOutOfRangeException	*rank* is invalid (being less than 1, for example).

The following member must be implemented if the Reflection library is present in the implementation.

Type.MakeByRefType() Method

```
[ILAsm]
.method public hidebysig virtual instance class System.Type
MakeByRefType()

[C#]
public virtual Type MakeByRefType()
```

Summary

Returns a `System.Type` object that represents the current type when passed as a byref parameter.

Return Value

A `System.Type` object that represents the current type when passed as a byref parameter.

Description

This method provides a way to generate a byref type for any type.

[*Note:* Using ilasm syntax, if the current `System.Type` object represents `int32`, this method returns a `System.Type` object representing `int32&`.]

The following member must be implemented if the Reflection library is present in the implementation.

Type.MakeGenericType(System.Type[]) Method

```
[ILAsm]
.method public hidebysig virtual class System.Type MakeGenericType(class
System.Type[] typeArguments)

[C#]
public override Type MakeGenericType(System.Type[] typeArguments)
```

Summary

Substitutes the elements of an array of types for the type parameters of the current generic type definition, and returns a `System.Type` object representing the resulting constructed type.

The current type shall be a generic type definition.

Parameters

Parameter	Description
typeArguments	An array of types to be substituted for the type parameters of the current generic type definition.

Return Value

A `System.Type` representing the constructed type formed by substituting the elements of *typeArguments* for the type parameters of the current generic type definition.

Description

This method allows you to write code that assigns specific types to the type parameters of a generic type definition, thus creating a `System.Type` object that represents a particular constructed type. You can use this `System.Type` object to create runtime instances of the constructed type.

The `System.Type` object returned by this method is the same as that obtained by calling the `System.Object.GetType` method of the resulting constructed type, or the `System.Object.GetType` method of any constructed type that was created from the same generic type using the same type arguments.

[*Note:* An array type whose element type is a generic type is not itself a generic type. Thus, you cannot call this method to bind an array type. To bind a type argument to this type, call `System.Type.GetElementType` to obtain the generic type, then this method to bind the type argument to the generic type, and, finally, `System.Type.MakeArrayType`

to create the array type.]

For a list of the invariant conditions for terms used in generic reflection, see the `System.Type.IsGenericType` property description.

Exceptions

Exception	Condition
System.ArgumentException	The number of elements in *typeArguments* is not the same as the number of type parameters of the current generic type definition. -or- An element of *typeArguments* does not satisfy the constraints specified for the corresponding type parameter of the current generic type definition.
System.ArgumentNullException	*typeArguments* is `null`. -or- An element of *typeArguments* is `null`.
System.InvalidOperationException	The current type does not represent the definition of a generic type. That is, `System.Type.IsGenericTypeDefinition` returns `false`.

Example

The following example uses `System.Type.GetType` and `System.Type.MakeGenericType` to create a constructed type from the generic `System.Collections.Generic.Dictionary` type. The constructed type represents a `System.Collections.Generic.Dictionary` of `Test` objects with string keys.

[C#]

```
using System;
using System.Reflection;
using System.Collections.Generic;

public class Test
{
```

```
        public static void Main()
        {
                Console.WriteLine("\n--- Create a constructed type from the
generic  Dictionary type.");

                // Create a type object representing the generic Dictionary
                // type.
                Type generic =
Type.GetType("System.Collections.Generic.Dictionary");

                DisplayTypeInfo(generic);

                // Create an array of types to substitute for the type
                // parameters of Dictionary. The key is of type string, and
                // the type to be contained in the Dictionary is Test.
                Type[] typeArgs = { typeof(string), typeof(Test) };
                Type constructed = generic.MakeGenericType(typeArgs);

                DisplayTypeInfo(constructed);

                // Compare the type objects obtained above to type objects
                // obtained using typeof() and GetGenericTypeDefinition().
                Console.WriteLine("\n--- Compare types obtained by different
methods:");

                Type t = typeof(Dictionary<string, Test>);

                Console.WriteLine("\tAre the constructed types equal? {0}", t
== constructed);
                Console.WriteLine("\tAre the generic types equal? {0}",
t.GetGenericTypeDefinition() == generic);
        }

        private static void DisplayTypeInfo(Type t)
        {
                Console.WriteLine("\n{0}", t);
                Console.WriteLine("\tIs this a generic type definition? {0}",
t.IsGenericTypeDefinition);
                Console.WriteLine("\tDoes it have generic type arguments?
{0}", t.HasGenericArguments);

                Type[] typeArguments = t.GetGenericArguments();

                Console.WriteLine("\tList type arguments ({0}):",
typeArguments.Length);
                foreach (Type tParam in typeArguments)
                {
                        Console.WriteLine("\t\t{0}", tParam);
                }
        }
}

/* This example produces the following output:

--- Create a constructed type from the generic Dictionary type.

System.Collections.Generic.Dictionary[KeyType,ValueType]
```

```
        Is this a generic type definition? True
        Does it have generic type arguments? True
        List type arguments (2):
                K
                V

System.Collections.Generic.Dictionary[System.String, Test]
        Is this a generic type definition? False
        Does it have generic type arguments? True
        List type arguments (2):
                System.String
                Test

--- Compare types obtained by different methods:
        Are the constructed types equal? True
        Are the generic types equal? True
 */

```

The following member must be implemented if the Reflection library is present in the implementation.

Type.MakePointerType() Method

```
[ILAsm]
.method public hidebysig virtual instance class System.Type
MakePointerType()

[C#]
public virtual Type MakePointerType()
```

Summary

Returns a `System.Type` object that represents the type of an unmanaged pointer to the current type.

Return Value

A `System.Type` object that represents the type of an unmanaged pointer to the current type.

Description

This method provides a way to generate an unmanaged pointer type for types computed at runtime.

[*Note:* Using ilasm syntax, if the current `System.Type` object represents `int32`, this method returns a `System.Type` object representing `int32*`.]

Type.ToString() Method

```
[ILAsm]
.method public hidebysig virtual string ToString()

[C#]
public override string ToString()
```

Summary

Returns a `System.String` representation of the current `System.Type`.

Return Value

Returns `System.Type.FullName`.

Description

[*Note:* This method overrides `System.Object.ToString`.]

If the current instance represents a generic type, the type and its type arguments are qualified by namespace and by nested type, but not by assembly. If the current instance represents an unassigned type parameter of a generic type or method, this method returns the unqualified name of the type parameter.

The following member must be implemented if the RuntimeInfrastructure library is present in the implementation.

Type.Assembly Property

```
[ILAsm]
.property class System.Reflection.Assembly Assembly { public hidebysig
virtual abstract specialname class System.Reflection.Assembly
get_Assembly() }

[C#]
public abstract Assembly Assembly { get; }
```

Summary

Gets the `System.Reflection.Assembly` in which the type is declared. For generic types, gets the `System.Reflection.Assembly` that contains the generic type definition.

Property Value

A `System.Reflection.Assembly` instance that describes the assembly containing the current type. For generic types, the instance describes the assembly that contains the definition of the generic type.

Description

If the current instance represents a generic type, this property returns the assembly in which the type was defined. For example, suppose you create an assembly named MyGenerics.dll that contains a class named `MyGenericStack<T>`. If you create an instance of `MyGenericStack<int>` in another assembly, the `System.Type.Assembly` property for the constructed type returns a `System.Reflection.Assembly` that represents MyGenerics.dll.

Similarly, if the current instance represents a generic parameter T, this property returns the assembly that contains the generic type definition that defines T.

Behaviors

This property is read-only.

The following member must be implemented if the Reflection library is present in the implementation.

Type.AssemblyQualifiedName Property

```
[ILAsm]
.property string AssemblyQualifiedName { public hidebysig virtual abstract
specialname string get_AssemblyQualifiedName() }

[C#]
public abstract string AssemblyQualifiedName { get; }
```

Summary

Gets the fully qualified name of the type represented by the current instance including the name of the assembly from which the `System.Type` was loaded.

Property Value

The assembly-qualified name of the `System.Type`, including the name of the assembly from which the `System.Type` was loaded. If the current `System.Type` object represents a generic parameter, this property returns `null`.

Behaviors

This property is read-only.

Compilers emit the simple name of a nested class, and reflection constructs a mangled name when queried, in accordance with the following conventions.

Delimiter	Meaning
Backslash (\)	Escape character.
Comma (,)	Precedes the Assembly name.
Plus sign (+)	Precedes a nested class.
Period (.)	Denotes namespace identifiers.
Square brackets ([])	After a type name, denotes an array of that type. -or- For a generic type, encloses the entire generic type argument list. -or-

	Within a type argument list, encloses an assembly-qualified type.

[*Note:* For example, the fully qualified name for a class might look like this:

TopNamespace.SubNameSpace.ContainingClass+NestedClass,MyAssembly

If the namespace were TopNamespace.Sub+Namespace, then the string would have to precede the plus sign (+) with an escape character (\) to prevent it from being interpreted as a nesting separator. Reflection emits this string as follows:

TopNamespace.Sub\+Namespace.ContainingClass+NestedClass,MyAssembly

A "++" becomes "\+\+", and a "\" becomes "\\".

]

Type names are permitted to include trailing characters that denote additional information about the type, such as whether the type is a reference type, a pointer type or an array type. To retrieve the type name without these trailing characters, use `t.GetElementType().ToString()`, where *t* is the type.

Spaces are significant in all type name components except the assembly name. In the assembly name, spaces before the ',' separator are significant, but spaces after the ',' separator are ignored.

Generic arguments of generic types are themselves fully qualified. For example, the output from the following C# program, if compiled to an assembly called Try64

```
using System;
using System.Reflection;

class MyTest {
  public static void Main(String[] args) {
    Type b = typeof(B<string,object>);
    Console.WriteLine(b.AssemblyQualifiedName);
  }
}
public class B<T,U> { }
```

is as follows:

```
B`2[[System.String, mscorlib, Version=2.0.3600.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089],[System.Object, mscorlib,
Version=2.0.3600.0, Culture=neutral, PublicKeyToken=b77a5c561934e089]],
Try64, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null
```

Usage

The name returned by this method can be persisted and later used to load the `System.Type`. To search for and load a `System.Type`, use `System.Type.GetType` either with the type name only or with the assembly qualified type name.

`System.Type.GetType` with the type name only will look for the `System.Type` in the caller's assembly and then in the System assembly. `System.Type.GetType` with the assembly qualified type name will look for the `System.Type` in any assembly.

The following member must be implemented if the Reflection library is present in the implementation.

Type.Attributes Property

```
[ILAsm]
.property valuetype System.Reflection.TypeAttributes Attributes { public
hidebysig specialname instance valuetype System.Reflection.TypeAttributes
get_Attributes() }

[C#]
public TypeAttributes Attributes { get; }
```

Summary

Gets the attributes associated with the type represented by the current instance.

Property Value

A `System.Reflection.TypeAttributes` object representing the attribute set of the `System.Type`.

Description

This property is read-only.

If the current instance represents a generic type, this property returns the attributes of the generic type definition.

If the current instance represents a generic type parameter T, the `System.Reflection.TypeAttributes` returned by this property includes `System.Reflection.TypeAttributes.Abstract`, `System.Reflection.TypeAttributes.AnsiClass`, `System.Reflection.TypeAttributes.AutoLayout`, `System.Reflection.TypeAttributes.Class`, `System.Reflection.TypeAttributes.Public`, and `System.Reflection.TypeAttributes.Sealed`. These are arbitrary choices which have no meaning in the context of a type parameter.

Type.BaseType Property

```
[ILAsm]
.property class System.Type BaseType { public hidebysig virtual abstract
specialname class System.Type get_BaseType() }

[C#]
public abstract Type BaseType { get; }
```

Summary

Gets the base `System.Type` of the current `System.Type`.

Property Value

A `System.Type` object representing the type from which the current `System.Type` directly inherits, or `null` if the current `System.Type` represents the `System.Object` class.

Description

The base type is the type from which the current type directly inherits. `System.Object` is the only type that does not have a base type, therefore `null` is returned as the base type of `System.Object`.

Interfaces inherit from `System.Object` and from zero or more base interfaces; therefore, the base type of an interface is considered to be `System.Object`.

If the current instance represents a constructed generic type, the base type reflects the generic arguments.

If the current instance represents an unassigned type parameter, `System.Type.BaseType` returns the base class type constraint declared for that parameter, or `System.Object` if no base class type constraint was declared.

Behaviors

This property is read-only.

Example

The following example demonstrates using the `System.Type.BaseType` property.

[C#]

```
using System;
class TestType {
 public static void Main() {
 Type t = typeof(int);
```

```
 Console.WriteLine("{0} inherits from {1}", t,t.BaseType);
 }
}
```

The output is

```
System.Int32 inherits from System.ValueType
```

The following member must be implemented if the Reflection library is present in the implementation.

Type.ContainsGenericParameters Property

```
[ILAsm]
.property bool ContainsGenericParameters { public hidebysig virtual
specialname bool get_ContainsGenericParameters() }

[C#]
public virtual bool ContainsGenericParameters { get; }
```

Summary

Gets a value that indicates whether a `System.Type` object contains unassigned generic parameters.

Property Value

`true` if a `System.Type` object contains unassigned generic parameters; otherwise `false`.

Description

In order to create an instance of a generic type, there must be no generic type definitions or open constructed types in the type arguments. For other constructed types, such as arrays and managed pointers, the types from which they are constructed must be able to be instantiated. If the `System.Type.ContainsGenericParameters` property returns `true`, the type cannot be instantiated.

The `System.Type.ContainsGenericParameters` property searches recursively for type parameters. For example, it returns true for an array whose element type is `A<T>`, even though the array type itself is not generic. Contrast this with the behavior of the `System.Type.IsGenericType` property, which returns `false` for arrays.

For a set of example classes and a table showing the values of the `System.Type.ContainsGenericParameters` property, see the `System.Type.IsGenericType` property description.

Behaviors

This property is read-only.

Example

For an example of using this method, see the example for `System.Type.GenericParameterPosition`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.DeclaringMethod Property

```
[ILAsm]
.property System.Reflection.MethodBase DeclaringMethod { public hidebysig
virtual specialname System.Reflection.MethodBase get_DeclaringMethod() }

[C#]
public virtual MethodBase DeclaringMethod { get; }
```

Summary

If the current `System.Type` represents a type parameter of a generic method, gets a `System.Reflection.MethodInfo` that represents the declaring method.

Property Value

If the current `System.Type` represents a type parameter of a generic method, a `System.Reflection.MethodInfo` that represents the declaring method; otherwise `null`.

Description

The declaring method is a generic method definition. That is, if `System.Type.DeclaringMethod` does not return `null`, then `DeclaringMethod.IsGenericMethodDefinition` returns `true`.

The `System.Type.DeclaringType` and `System.Type.DeclaringMethod` properties identify the generic type definition or generic method definition where the generic type parameter was originally defined:

- If the `System.Type.DeclaringMethod` property returns a `System.Reflection.MethodBase`, that `System.Reflection.MethodBase` represents a generic method definition, and the current `System.Type` object represents a type parameter of that generic method definition.

- If the `System.Type.DeclaringMethod` property returns a `null`, then the `System.Type.DeclaringType` property always returns a `System.Type` object representing a generic type definition, and the current `System.Type` object represents a type parameter of that generic type definition.

For a list of the invariant conditions for terms used in generic reflection, see the `System.Type.IsGenericType` property description.

Behaviors

This property is read-only.

The following member must be implemented if the Reflection library is present in the implementation.

Type.DeclaringType Property

```
[ILAsm]
.property class System.Type DeclaringType { public hidebysig virtual
specialname class System.Type get_DeclaringType() }

[C#]
public virtual Type DeclaringType { get; }
```

Summary

Gets the type that declares the type represented by the current instance.

Property Value

The `System.Type` object for the class that declares the type represented by the current instance. If the type is a nested type, this property returns the enclosing type; otherwise, returns the current instance.

Description

[*Note:* This property implements the abstract property inherited from `System.Reflection.MemberInfo`.]

If the current `System.Type` represents a type parameter of a generic type or method definition, the `System.Type.DeclaringType` and `System.Type.DeclaringMethod` properties identify the generic type definition or generic method definition where the generic type parameter was originally defined:

- If the `System.Type.DeclaringMethod` property returns a `System.Reflection.MethodBase`, that `System.Reflection.MethodBase` represents a generic method definition, and the current `System.Type` object represents a type parameter of that generic method definition.

- If the `System.Type.DeclaringMethod` property returns a `null`, then the `System.Type.DeclaringType` property always returns a `System.Type` object representing a generic type definition, and the current `System.Type` object represents a type parameter of that generic type definition.

For a type parameter of a generic method, this property returns the type that contains the generic method definition.

The following member must be implemented if the Reflection library is present in the implementation.

Type.DefaultBinder Property

```
[ILAsm]
.property class System.Reflection.Binder DefaultBinder { public hidebysig
static specialname class System.Reflection.Binder get_DefaultBinder() }

[C#]
public static Binder DefaultBinder { get; }
```

Summary

Gets the default binder used by the system.

Property Value

The default `System.Reflection.Binder` used by the system.

Description

This property is read-only.

Reflection models the accessibility rules of the common type system. For example, if the caller is in the same assembly, the caller does not need special permissions for internal members. Otherwise, the caller needs `System.Security.Permissions.ReflectionPermission`. This is consistent with lookup of members that are protected, private, and so on.

[*Note:* The general principle is that `System.Reflection.Binder.ChangeType` typically performs only widening coercions, which never lose data. An example of a widening coercion is coercing a value that is a 32-bit signed integer to a value that is a 64-bit signed integer. This is distinguished from a narrowing coercion, which can lose data. An example of a narrowing coercion is coercing a 64-bit signed integer to a 32-bit signed integer.]

The following table lists the coercions performed by the default binder's implementation of `ChangeType`.

Source Type	Target Type
Any type	Its base type.
Any type	The interface it implements.
Char	Unt16, UInt32, Int32, UInt64, Int64, Single, Double

Byte	Char, Unt16, Int16, UInt32, Int32, UInt64, Int64, Single, Double
SByte	Int16, Int32, Int64, Single, Double
UInt16	UInt32, Int32, UInt64, Int64, Single, Double
Int16	Int32, Int64, Single, Double
UInt32	UInt64, Int64, Single, Double
Int32	Int64, Single, Double
UInt64	Single, Double
Int64	Single, Double
Single	Double
Non-reference	By-reference.

1

2

Type.FullName Property

[ILAsm]
```
.property string FullName { public hidebysig virtual abstract specialname
string get_FullName() }
```

[C#]
```
public abstract string FullName { get; }
```

Summary

Gets the fully qualified name of the type represented by the current instance.

Property Value

A `System.String` containing the fully qualified name of the `System.Type`.

Description

[*Note:* For example, the fully qualified name of the C# string type is "System.String".]

If the current instance represents a generic type, the type arguments in the string returned by `System.Type.FullName` are qualified by their assembly, version, and so on, even though the string representation of the generic type itself is not qualified by assembly. Thus, `t.FullName + ", " + t.Assembly.FullName` produces the same result as `t.AssemblyQualifiedName`, as with types that are not generic.

If the current instance represents an unassigned type parameter of a generic type, this property returns `null`.

Behaviors

This property is read-only.

Example

The following example demonstrates using the `System.Type.FullName` property.

[C#]

```
using System;
class TestType {
 public static void Main() {
 Type t = typeof(Array);
 Console.WriteLine("Full name of Array type is {0}",t.FullName);
 }
}
```

```
1   The output is
2
3   Full name of Array type is System.Array

4
```

Type.GenericParameterAttributes Property

```
[ILAsm]
.property valuetype System.Reflection.GenericParameterAttributes
GenericParameterAttributes { public hidebysig specialname virtual instance
valuetype System.Reflection.GenericParameterAttributes
get_GenericParameterAttributes () }

[C#]
public virtual GenericParameterAttributes GenericParameterAttributes {
get; }
```

Summary

Gets a combination of `System.Reflection.GenericParameterAttributes` flags that describe the variance and special constraints of the current generic type parameter.

Property Value

A `System.Reflection.GenericParameterAttributes` value that describes the variance and special constraints of the current generic type parameter.

Description

This property is read-only.

The value of this property contains flags that describe whether the current generic type parameter is variant, and flags that describe any special constraints. Use `System.GenericParameterAttributes.VarianceMask` to select the variance flags, and `System.GenericParameterAttributes.SpecialConstraintMask` to select the constraint flags. Use `System.Reflection.GetGenericParameterConstraints` to get the type constraints.

For a list of the invariant conditions for terms used in generic reflection, see the `System.Type.IsGenericType` property description.

Exceptions

Exception	Condition
System.InvalidOperationException	The current `System.Type` object is not a generic type parameter. That is, the `System.Type.IsGenericParameter` property returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.GenericParameterPosition Property

```
[ILAsm]
.property int GenericParameterPosition { public hidebysig virtual
specialname int get_GenericParameterPosition() }

[C#]
public virtual int GenericParameterPosition { get; }
```

Summary

For a `System.Type` object that represents a type parameter of a generic type or generic method, gets the position of the type parameter in the type parameter list of the generic type or generic method.

Property Value

A zero-based integer representing the position of a type parameter in the type parameter list of the generic type or generic method that declared the parameter.

Description

This read-only property returns the position of a type parameter in the parameter list of the generic type definition or generic method definition where the type parameter was originally defined. The `System.Type.DeclaringType` and `System.Type.DeclaringMethod` properties identify the generic type or method definition:

- If the `System.Type.DeclaringMethod` property returns a `System.Reflection.MethodBase`, that `System.Reflection.MethodBase` represents a generic method definition, and the current `System.Type` object represents a type parameter of that generic method definition.

- If the `System.Type.DeclaringMethod` property returns a `null`, then the `System.Type.DeclaringType` property always returns a `System.Type` object representing a generic type definition, and the current `System.Type` object represents a type parameter of that generic type definition.

To provide the correct context for the value of the `System.Type.GenericParameterPosition` property, it is necessary to identify the generic type or method a type parameter belongs to. For example, consider the return value of the generic method `GetSomething` in the following C# code:

```
public class B<T, U> { }
 public class A<V>
 {
     public B<V, X> GetSomething<X>()
     {
```

```
            return new Base<V, X>();
        }
    }
```

The type returned by `GetSomething` depends on the type arguments supplied to class `A` and `GetSomething` itself. You can obtain a `System.Reflection.MethodInfo` for `GetSomething` and from that you can obtain the return type. When you examine the type parameters of the return type, `System.Type.GenericParameterPosition` returns zero for both. The position of `V` is zero because `V` is the first type parameter in the type parameter list for class `A`. The position of`X` is zero because `X` is the first type parameter in the type parameter list for `GetSomething`.

[*Note:* Calling the `System.Type.GenericParameterPosition` property causes an exception if the current `System.Type` does not represent a type parameter. When you examine the type arguments of an open constructed type, use the `System.Type.IsGenericParameter` property to tell which are type parameters and which are types. The `System.Type.IsGenericParameter` property returns`true` for a type parameter; you can then use the `System.Type.GenericParameterPosition` method to obtain its position, and the `System.Type.DeclaringMethod` and `System.Type.DeclaringType` properties to determine the generic method or type definition that defines it.

]

Exceptions

Exception	Condition
System.InvalidOperationException	The current type does not represent a type parameter. That is, `System.Type.IsGenericParameter` returns `false`.

Example

The following example defines a generic class with two type parameters, and a generic class that derives from it. The base class of the derived type has one unbound type parameter and one type parameter bound to `System.Int32`. The example displays information about these generic classes, including the positions reported by `System.Type.GenericParameterPosition`.

[C#]

```
using System;
using System.Reflection;
using System.Collections.Generic;
// Define a base class with two type parameters.
public class Base<T, U> { }

// Define a derived class. The derived class inherits from a constructed
// class that meets the following criteria:
```

```
//   (1) Its generic type definition is Base<T, U>.
//   (2) It specifies int for the first type parameter.
//   (3) For the second type parameter, it uses the same type that is used
//       for the type parameter of the derived class.
// Thus, the derived class is a generic type with one type parameter, but
// its base class is an open constructed type with one type argument and
// one type parameter.
public class Derived<V>: Base<int,V> { }

public class Test
{
        public static void Main()
        {
                Console.WriteLine("\n--- Display a generic type and the open
constructed");
                Console.WriteLine("    type from which it is derived.");

                // Create a Type object representing the generic type Derived.
                //
                Type derivedType = Type.GetType("Derived");

                DisplayGenericTypeInfo(derivedType);

                // Display its open constructed base type.
                DisplayGenericTypeInfo(derivedType.BaseType);
        }

        private static void DisplayGenericTypeInfo(Type t)
        {
                Console.WriteLine("\n{0}", t);
                Console.WriteLine("\tIs this a generic type definition? {0}",
t.IsGenericTypeDefinition);
                Console.WriteLine("\tDoes it have generic arguments? {0}",
t.HasGenericArguments);
                Console.WriteLine("\tDoes it have unbound generic parameters?
{0}", t.ContainsGenericParameters);
                if (t.HasGenericArguments)
                {
                        // If the type is a generic type definition or a
                        // constructed type, display the type arguments.
                        //
                        Type[] typeArguments = t.GetGenericArguments();

                        Console.WriteLine("\tList type arguments ({0}):",
typeArguments.Length);
                        foreach (Type tParam in typeArguments)
                        {
                                // IsGenericParameter is true only for generic
type
                                // parameters.
                                //
                                if (tParam.IsGenericParameter)
                                {
                                        Console.WriteLine("\t\t{0}  (unbound -
parameter position {1})", tParam, tParam.GenericParameterPosition);
                                }
                                else
```

```
                                {
                                        Console.WriteLine("\t\t{0}", tParam);
                                }
                        }
                }
                else
                {
                        Console.WriteLine("\tThis is not a generic or
constructed type.");
                }
        }
}

/* This example produces the following output:

--- Display a generic type and the open constructed
    type from which it is derived.

Derived[V]
        Is this a generic type definition? True
        Does it have generic arguments? True
        Does it have unbound generic parameters? True
        List type arguments (1):
                V  (unbound - parameter position 0)

Base[System.Int32, V]
        Is this a generic type definition? False
        Does it have generic arguments? True
        Does it have unbound generic parameters? True
        List type arguments (2):
                System.Int32
                V  (unbound - parameter position 0)
 */

```

The following member must be implemented if the Reflection library is present in the implementation.

Type.HasElementType Property

```
[ILAsm]
.property bool HasElementType { public hidebysig specialname instance bool
get_HasElementType() }

[C#]
public bool HasElementType { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the type represented by the current instance encompasses or refers to another type; that is, whether the current `System.Type` is an array, a pointer, or is passed by reference.

Property Value

`true` if the `System.Type` is an array, a pointer, or is passed by reference; otherwise, `false`.

Description

This property is read-only.

[*Note:* For example, `System.Type.GetType`("`System.Int32` []").HasElementType returns `true`, but `System.Type.GetType`("`System.Int32` ").HasElementType returns `false`. `System.Type.HasElementType` also returns `true` for "Int32*" and "Int32&".]

If the current instance represents a generic type, or a type parameter of a generic type or method, this property returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsAbstract Property

```
[ILAsm]
.property bool IsAbstract { public hidebysig specialname instance bool
get_IsAbstract() }

[C#]
public bool IsAbstract { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the type represented by the current instance is abstract and is required to be overridden.

Property Value

`true` if the `System.Type` is abstract; otherwise, `false`.

Description

This property is read-only.

If the current instance represents an unassigned type parameter of a generic type, this property always returns `true`. This is because it is not possible to create an instance of a generic type parameter.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsAnsiClass Property

```
[ILAsm]
.property bool IsAnsiClass { public hidebysig specialname virtual instance
bool get_IsAnsiClass() }

[C#]
public virtual bool IsAnsiClass { get; }
```

Summary

Indicates whether the type attribute `System.Reflection.TypeAttributes.AnsiClass` is selected for the current type.

Property Value

`true` if the type attribute `System.Reflection.TypeAttributes.AnsiClass` is selected for the current type; otherwise, `false`.

Description

This property is read-only.

If the current `System.Type` represents a generic type, this property applies to the definition of the type. If the current `System.Type` represents a type parameter of a generic type or method, this property always returns `false`.

Type.IsArray Property

```
[ILAsm]
.property bool IsArray { public hidebysig specialname instance bool
get_IsArray() }

[C#]
public bool IsArray { get; }
```

Summary

Gets a `System.Boolean` value that indicates whether the current `System.Type` represents an array.

Property Value

`true` if the current `System.Type` represents an array; otherwise `false`.

Description

This property is read-only.

This property returns `true` for an array of objects, but not for the `System.Array` type itself, which is a class.

If the current instance represents a generic type, or a type parameter of a generic type or method, this property returns `false`.

Example

The following example demonstrates using the `System.Type.IsArray` property.

[C#]

```
using System;
class TestType {
 public static void Main() {
 int [] array = {1,2,3,4};
 Type at = typeof(Array);
 Type t = array.GetType();
 Console.WriteLine("Type is {0}. IsArray? {1}", at, at.IsArray);
 Console.WriteLine("Type is {0}. IsArray? {1}", t, t.IsArray);
 }
}
```

The output is

```
Type is System.Array. IsArray? False


Type is System.Int32[]. IsArray? True
```

1

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsAutoClass Property

```
[ILAsm]
.property bool IsAutoClass { public hidebysig specialname virtual instance
bool get_IsAutoClass() }

[C#]
public virtual bool IsAutoClass { get; }
```

Summary

Indicates whether the type attribute `System.Reflection.TypeAttributes.AutoClass` is selected for the current type.

Property Value

`true` if the type attribute `System.Reflection.TypeAttributes.AutoClass` is selected for the current type; otherwise, `false`.

Description

This property is read-only.

If the current `System.Type` represents a generic type, this property applies to the definition of the type. If the current `System.Type` represents a type parameter of a generic type or method, this property always returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsAutoLayout Property

```
[ILAsm]
.property bool IsAutoLayout { public hidebysig specialname instance bool
get_IsAutoLayout() }

[C#]
public bool IsAutoLayout { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the type layout attribute `System.Reflection.TypeAttributes.AutoLayout` is specified for the `System.Type`.

Property Value

`true` if the type layout attribute `System.Reflection.TypeAttributes.AutoLayout` is specified for the current `System.Type`; otherwise, `false`.

Description

This property is read-only.

If the current instance represents a generic type, this property applies to the definition of the type. If the current instance represents an unassigned type parameter of a generic type or method, this property always returns `false`.

[*Note:* The `System.Reflection.TypeAttributes.AutoLayout` attribute specifies that the system selects the layout the objects of the type. Types marked with this attribute indicate that the system will choose the appropriate way to lay out the type; any layout information that might have been specified is ignored.

]

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsByRef Property

```
[ILAsm]
.property bool IsByRef { public hidebysig specialname instance bool
get_IsByRef() }

[C#]
public bool IsByRef { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the `System.Type` is passed by reference.

Property Value

`true` if the `System.Type` is passed by reference; otherwise, `false`.

Description

This property is read-only.

Type.IsClass Property

```
[ILAsm]
.property bool IsClass { public hidebysig specialname instance bool
get_IsClass() }

[C#]
public bool IsClass { get; }
```

Summary

Gets a `System.Boolean` value that indicates whether the current `System.Type` represents a class.

Property Value

`true` if the current `System.Type` represents a class; otherwise `false`.

Description

This property is read-only.

Note that this property returns `true` for `System.Type` instances representing `System.Enum` and `System.ValueType`.

If the current instance represents a generic type, this property returns `true` if the generic type definition is a class definition (that is, it does not define an interface or a value type).

If the current instance represents an unassigned type parameter of a generic type or method, this property always returns `false`.

Type.IsEnum Property

```
[ILAsm]
.property bool IsEnum { public hidebysig specialname instance bool
get_IsEnum() }

[C#]
public bool IsEnum { get; }
```

Summary

Gets a `System.Boolean` value that indicates whether the current `System.Type` represents an enumeration.

Property Value

`true` if the current `System.Type` represents an enumeration; otherwise `false`.

Description

This property is read-only.

This property returns `true` for an enumeration, but not for the `System.Enum` type itself, which is a class.

If the current instance represents a generic type, this property applies to the definition of the type. If the current instance represents an unassigned type parameter of a generic type or method, this property always returns `false`.

Example

The following example demonstrates using the `System.Type.IsEnum` property.

[C#]

```
using System;
public enum Color {
Red, Blue, Green
}
class TestType {
 public static void Main() {
 Type colorType = typeof(Color);
 Type enumType = typeof(Enum);
 Console.WriteLine("Color is enum ? {0}", colorType.IsEnum);
 Console.WriteLine("Color is valueType? {0}", colorType.IsValueType);
 Console.WriteLine("Enum is enum Type? {0}", enumType.IsEnum);
 Console.WriteLine("Enum is value? {0}", enumType.IsValueType);
 }
}
```

The output is

```
Color is enum ? True
```

```
 1
 2
 3   Color is valueType? True
 4
 5
 6   Enum is enum Type? False
 7
 8
 9   Enum is value? False
10
11
```

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsExplicitLayout Property

```
[ILAsm]
.property bool IsExplicitLayout { public hidebysig specialname instance
bool get_IsExplicitLayout() }

[C#]
public bool IsExplicitLayout { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the type layout attribute `System.Reflection.TypeAttributes.ExplicitLayout` is specified for the `System.Type`.

Property Value

`true` if the type layout attribute `System.Reflection.TypeAttributes.ExplicitLayout` is specified for the current `System.Type`; otherwise, `false`.

Description

This property is read-only.

[*Note:* Types marked with the `System.Reflection.TypeAttributes.ExplicitLayout` attribute cause the system to ignore field sequence and to use the explicit layout rules provided, in the form of field offsets, overall class size and alignment.

]

If the current instance represents a generic type, this property applies to the definition of the type. If the current instance represents an unassigned type parameter of a generic type or method, this property always returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsGenericParameter Property

```
[ILAsm]
.property bool IsGenericParameter { public virtual hidebysig specialname
instance bool get_IsGenericParameter() }

[C#]
public virtual bool IsGenericParameter { get; }
```

Summary

Gets a value that indicates whether the current type represents a type parameter of a generic type or method.

Property Value

`true` if the current object represents a type parameter of a generic type or method; otherwise `false`.

Description

This property is read-only.

Use this property to distinguish between type parameters and type arguments. When you call `System.Type.GetGenericArguments` to obtain the type arguments of a generic type, some elements of the array might be specific types (type arguments) and others might be type parameters.`System.Type.IsGenericParameter` returns `false` for the types and `true` for the type parameters.

For a list of the invariant conditions for terms used in generic reflection, see the `System.Type.IsGenericType` property description.

Example

For an example of using this method, see the example for `System.Type.GenericParameterPosition`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsGenericType Property

```
[ILAsm]
.property bool IsGenericType { public hidebysig virtual specialname bool
get_IsGenericType() }

[C#]
public virtual bool IsGenericType { get; }
```

Summary

Gets a value that indicates whether the current type has type arguments, and is therefore a generic type.

Property Value

`true` if the current type has type arguments; otherwise `false`.

Description

Use this property to determine whether a `System.Type` object represents a generic type. Use the `System.Type.ContainsGenericParameters` property to determine whether a `System.Type` object represents an open constructed type or a closed constructed type.

[*Note:* The `System.Type.HasGenericArguments` property returns `false` if the immediate type is not generic.]

The following table summarizes the invariant conditions for common terms used in generic reflection.

Term	Invariant
generic type definition	The `System.Type.IsGenericTypeDefinition` property is `true`. Defines a generic type. A constructed type is created by calling the `System.Type.MakeGenericType(System.Type[])` method on a `System.Type` object that represents a generic type definition, and specifying an array of type arguments. `System.Type.MakeGenericType(System.Type[])` can be called only on generic type definitions. Any generic type definition is a generic type, but the converse is not true.

generic type	The `System.Type.IsGenericType` property is `true`. Can be a generic type definition, an open constructed type, or a closed constructed type. Note that an array type whose element type is generic is not itself a generic type. The same is true of a `System.Type` object representing a pointer to a generic type.
open constructed type	The `System.Type.ContainsGenericParameters` property is `true`. It is not possible to create an instance of an open constructed type. Note that not all open constructed types are generic, such as an array type whose element type is a generic type definition.
closed constructed type	The `System.Type.ContainsGenericParameters` property is `false`. When examined recursively, the type has no unassigned generic parameters. The containing type or method has no generic type parameters, and, recursively, no type arguments have unassigned generic type parameters.
generic type parameter	The `System.Type.IsGenericParameter` property is `true`. In a generic type definition, a placeholder for a type that will be assigned later.
generic type argument	Can be any type, including a generic type parameter. Type arguments are specified as an array of `System.Type` objects passed to the `System.Type.MakeGenericType(System.Type[])` method when creating a constructed generic type. If instances of the resulting type are to be created, the `System.Type.ContainsGenericParameters` property must be `false` for all the type arguments.

Behaviors

This property is read-only.

Example

For an example of using this method, see the example for `System.Type.MakeGenericType`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsGenericTypeDefinition Property

```
[ILAsm]
.property virtual bool IsGenericTypeDefinition { public hidebysig
specialname instance bool get_IsGenericTypeDefinition() }

[C#]
public virtual bool IsGenericTypeDefinition { get; }
```

Summary

Gets a value that indicates whether the current object represents the definition of a generic type, or whether one or more of its type parameters has been specified.

Property Value

`true` if the current object represents the definition of a generic type, none of whose type parameters have been bound to specific types; otherwise `false`.

Description

This property is read-only.

Use this property to determine whether type arguments have been specified for any of the type parameters of a generic type. If type arguments have been specified (that is, bound to the corresponding type parameters), this property returns `false`.

For a list of the invariant conditions for terms used in generic reflection, see the `System.Type.IsGenericType` property description.

[*Note:* An open generic type can have type parameters even if types have been specified for its type parameters.

]

Example

For an example of using this method, see the example for `System.Type.MakeGenericType`..

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsImport Property

```
[ILAsm]
.property bool IsImport { public hidebysig specialname instance bool
get_IsImport() }

[C#]
public bool IsImport { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the `System.Type` was imported from another class.

Property Value

`true` if the `System.Type` was imported from another class; otherwise, `false`.

Description

This property is read-only.

If the current instance represents a generic type, this property applies to the definition of the type. If the current instance represents an unassigned type parameter of a generic type or method, this property always returns `false`.

Type.IsInterface Property

```
[ILAsm]
.property bool IsInterface { public hidebysig specialname instance bool
get_IsInterface() }

[C#]
public bool IsInterface { get; }
```

Summary

Gets a `System.Boolean` value that indicates whether the current `System.Type` represents an interface.

Property Value

`true` if the current `System.Type` represents an interface; otherwise `false`.

Description

This property is read-only.

If the current instance represents an unassigned type parameter of a generic type or method, this property always returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsLayoutSequential Property

```
[ILAsm]
.property bool IsLayoutSequential { public hidebysig specialname instance
bool get_IsLayoutSequential() }

[C#]
public bool IsLayoutSequential { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the type layout attribute `System.Reflection.TypeAttributes.SequentialLayout` is specified for the `System.Type`.

Property Value

`true` if the type layout attribute `System.Reflection.TypeAttributes.SequentialLayout` is specified for the current `System.Type`; otherwise, `false`.

Description

This property is read-only.

[*Note:* The `System.Reflection.TypeAttributes.SequentialLayout` attribute is used to indicate that the system is to preserve field order as emitted, but otherwise the specific offsets are calculated based on the type of the field; these might be shifted by explicit offset, padding, or alignment information.

]

If the current instance represents a generic type, this property applies to the definition of the type. If the current instance represents an unassigned type parameter of a generic type or method, this property always returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsMarshalByRef Property

```
[ILAsm]
.property bool IsMarshalByRef { public hidebysig specialname instance bool
get_IsMarshalByRef() }

[C#]
public bool IsMarshalByRef { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current type is marshaled by reference.

Property Value

`true` if the `System.Type` is marshaled by reference; otherwise, `false`.

Description

This property is read-only.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsNestedAssembly Property

```
[ILAsm]
.property bool IsNestedAssembly { public hidebysig specialname instance
bool get_IsNestedAssembly() }

[C#]
public bool IsNestedAssembly { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current `System.Type` is nested and visible only within its own assembly.

Property Value

`true` if the `System.Type` is nested and visible only within its own assembly; otherwise, `false`.

Description

This property is read-only.

If the current instance represents an unassigned type parameter of a generic type, this property returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsNestedFamANDAssem Property

```
[ILAsm]
.property bool IsNestedFamANDAssem { public hidebysig specialname instance
bool get_IsNestedFamANDAssem() }

[C#]
public bool IsNestedFamANDAssem { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current `System.Type` is nested and visible only to classes that belong to both its own family and its own assembly.

Property Value

`true` if the `System.Type` is nested and visible only to classes that belong to both its own family and its own assembly; otherwise, `false`.

Description

This property is read-only.

A `System.Type` object's family is defined as all objects of the exact same `System.Type` and of its subclasses.

If the current instance represents an unassigned type parameter of a generic type, this property returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsNestedFamily Property

```
[ILAsm]
.property bool IsNestedFamily { public hidebysig specialname instance bool
get_IsNestedFamily() }

[C#]
public bool IsNestedFamily { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current `System.Type` is nested and visible only within its own family.

Property Value

`true` if the `System.Type` is nested and visible only within its own family; otherwise, `false`.

Description

This property is read-only.

A `System.Type` object's family is defined as all objects of the exact same `System.Type` and of its subclasses.

If the current instance represents an unassigned type parameter of a generic type, this property returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsNestedFamORAssem Property

```
[ILAsm]
.property bool IsNestedFamORAssem { public hidebysig specialname instance
bool get_IsNestedFamORAssem() }

[C#]
public bool IsNestedFamORAssem { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current `System.Type` is nested and visible only to classes that belong to either its own family or to its own assembly.

Property Value

`true` if the `System.Type` is nested and visible only to classes that belong to its own family or to its own assembly; otherwise, `false`.

Description

This property is read-only.

A `System.Type` object's family is defined as all objects of the exact same `System.Type` and of its subclasses.

If the current instance represents an unassigned type parameter of a generic type, this property returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsNestedPrivate Property

```
[ILAsm]
.property bool IsNestedPrivate { public hidebysig specialname instance
bool get_IsNestedPrivate() }

[C#]
public bool IsNestedPrivate { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current `System.Type` is nested and declared private.

Property Value

`true` if the `System.Type` is nested and declared private; otherwise, `false`.

Description

This property is read-only.

If the current instance represents an unassigned type parameter of a generic type, this property returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsNestedPublic Property

```
[ILAsm]
.property bool IsNestedPublic { public hidebysig specialname instance bool
get_IsNestedPublic() }

[C#]
public bool IsNestedPublic { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current `System.Type` is a public nested class.

Property Value

`true` if the class is nested and declared public; otherwise, `false`.

Description

This property is read-only.

If the current instance represents an unassigned type parameter of a generic type, this property returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsNotPublic Property

```
[ILAsm]
.property bool IsNotPublic { public hidebysig specialname instance bool
get_IsNotPublic() }

[C#]
public bool IsNotPublic { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the top-level `System.Type` is not declared public.

Property Value

`true` if the top-level `System.Type` is not declared public; otherwise, `false`.

Description

This property is read-only.

If the current instance represents an unassigned type parameter of a generic type, this property returns `false`.

Type.IsPointer Property

```
[ILAsm]
.property bool IsPointer { public hidebysig specialname instance bool
get_IsPointer() }

[C#]
public bool IsPointer { get; }
```

Summary

Gets a `System.Boolean` value that indicates whether the current `System.Type` represents a pointer.

Property Value

This property is read-only.

`true` if the current `System.Type` represents a pointer; otherwise `false`.

Description

This property is read-only.

If the current instance represents a generic type, or a type parameter of a generic type or method, this property always returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsPrimitive Property

```
[ILAsm]
.property bool IsPrimitive { public hidebysig specialname instance bool
get_IsPrimitive() }

[C#]
public bool IsPrimitive { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current `System.Type` is one of the primitive types.

Property Value

`true` if the `System.Type` is one of the primitive types; otherwise, `false`.

Description

This property is read-only.

The primitive types are `System.Boolean`, `System.Byte`, `System.SByte`, `System.Int16`, `System.UInt16`, `System.Int32`, `System.UInt32`, `System.Int64`, `System.UInt64`, `System.Char`, `System.Double`, and `System.Single`.

If the current instance represents a generic type, or a type parameter of a generic type or method, this property always returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsPublic Property

```
[ILAsm]
.property bool IsPublic { public hidebysig specialname instance bool
get_IsPublic() }

[C#]
public bool IsPublic { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the top-level `System.Type` is declared public.

Property Value

`true` if the top-level `System.Type` is declared public; otherwise, `false`.

Description

This property is read-only.

If the current instance represents an unassigned type parameter of a generic type, this property returns `true`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsSealed Property

```
[ILAsm]
.property bool IsSealed { public hidebysig specialname instance bool
get_IsSealed() }

[C#]
public bool IsSealed { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current `System.Type` is declared sealed.

Property Value

`true` if the `System.Type` is declared sealed; otherwise, `false`.

Description

This property is read-only.

If the current instance represents an unassigned type parameter of a generic type, this property returns `true`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsSpecialName Property

```
[ILAsm]
.property bool IsSpecialName { public hidebysig specialname instance bool
get_IsSpecialName() }

[C#]
public bool IsSpecialName { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current `System.Type` has a name that requires special handling.

Property Value

`true` if the `System.Type` has a name that requires special handling; otherwise, `false`.

Description

This property is read-only.

[*Note:* Names that begin with or contain an underscore character (_) are examples of type names that might require special treatment by some tools.]

If the current instance represents a generic type, this property applies to the definition of the type. If the current instance represents an unassigned type parameter of a generic type or method, this property always returns `false`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.IsUnicodeClass Property

```
[ILAsm]
.property bool IsUnicodeClass { public hidebysig specialname virtual
instance bool get_IsUnicodeClass() }

[C#]
public virtual bool IsUnicodeClass { get; }
```

Summary

Indicates whether the type attribute `System.Reflection.TypeAttributes.UnicodeClass` is selected for the current type.

Property Value

`true` if the type attribute `System.Reflection.TypeAttributes.UnicodeClass` is selected for the current type; otherwise, `false`.

Description

This property is read-only.

If the current `System.Type` represents a generic type, this property applies to the definition of the type. If the current `System.Type` represents a type parameter of a generic type or method, this property always returns `false`.

Type.IsValueType Property

```
[ILAsm]
.property bool IsValueType { public hidebysig specialname instance bool
get_IsValueType() }

[C#]
public bool IsValueType { get; }
```

Summary

Gets a `System.Boolean` value that indicates whether the current `System.Type` represents a value type.

Property Value

`true` if the current `System.Type` represents a value type (structure); otherwise `false`.

Description

This property is read-only.

This property returns true for enumerations, but not for the `System.Enum` type itself, which is a class. [*Note:* For an example that demonstrates this behavior, see `System.Type.IsEnum`.]

The following member must be implemented if the Reflection library is present in the implementation.

Type.Module Property

```
[ILAsm]
.property class System.Reflection.Module Module { public hidebysig virtual
abstract specialname class System.Reflection.Module get_Module() }

[C#]
public abstract Module Module { get; }
```

Summary

Gets the module in which the current `System.Type` is defined.

Property Value

A `System.Reflection.Module` that reflects the module in which the current `System.Type` is defined.

Description

If the current instance represents a generic type, this property returns the module in which the type was defined.

Similarly, if the current instance represents a generic parameter T, this property returns the assembly that contains the generic type that defines T.

Behaviors

This property is read-only.

The following member must be implemented if the Reflection library is present in the implementation.

Type.Namespace Property

```
[ILAsm]
.property string Namespace { public hidebysig virtual abstract specialname
string get_Namespace() }

[C#]
public abstract string Namespace { get; }
```

Summary

Gets the namespace of the `System.Type`.

Property Value

A `System.String` containing the namespace of the current `System.Type`.

Description

If the current instance represents a generic type, this property returns the namespace that contains the generic type definition. Similarly, if the current instance represents a generic parameter T, this property returns the namespace that contains the generic type that defines T.

[*Note:* A namespace is a logical design-time naming convenience, used mainly to define scope in an application and organize classes and other types in a hierarchical structure. From the viewpoint of the system, there are no namespaces.]

Behaviors

This property is read-only.

The following member must be implemented if the Reflection library is present in the implementation.

Type.ReflectedType Property

```
[ILAsm]
.property class System.Type ReflectedType { public hidebysig virtual
specialname class System.Type get_ReflectedType() }

[C#]
public override Type ReflectedType { get; }
```

Summary

Gets the type that was used to obtain the current instance.

Property Value

The `Type` object through which the current instance was obtained.

Description

This property is read-only.

If the current instance represents a generic type, or a type parameter of a generic type or method, this property returns the current instance.

The following member must be implemented if the RuntimeInfrastructure library is present in the implementation.

Type.TypeHandle Property

```
[ILAsm]
.property valuetype System.RuntimeTypeHandle TypeHandle { public hidebysig
virtual abstract specialname valuetype System.RuntimeTypeHandle
get_TypeHandle() }

[C#]
public abstract RuntimeTypeHandle TypeHandle { get; }
```

Summary

Gets the handle for the current `System.Type`.

Property Value

The `System.RuntimeTypeHandle` for the current `System.Type`.

Description

This property is read-only.

The `System.RuntimeTypeHandle` encapsulates a pointer to an internal data structure that represents the type. This handle is unique during the process lifetime. The handle is valid only in the application domain in which it was obtained.

The following member must be implemented if the Reflection library is present in the implementation.

Type.TypeInitializer Property

```
[ILAsm]
.property class System.Reflection.ConstructorInfo TypeInitializer { public
hidebysig specialname instance class System.Reflection.ConstructorInfo
get_TypeInitializer() }

[C#]
public ConstructorInfo TypeInitializer { get; }
```

Summary

Gets the initializer for the type represented by the current instance.

Property Value

A `System.Reflection.ConstructorInfo` containing the name of the static constructor for the type represented by the current instance

Description

This property is read-only.

[*Note:* Type initializers are available through `System.Type.GetMember`, `System.Type.GetMembers`, and `System.Type.GetConstructors`.]

If the current instance represents an unassigned type parameter of a generic type or method, this property returns `null`.

The following member must be implemented if the Reflection library is present in the implementation.

Type.UnderlyingSystemType Property

```
[ILAsm]
.property class System.Type UnderlyingSystemType { public hidebysig
virtual abstract specialname class System.Type get_UnderlyingSystemType()
}

[C#]
public abstract Type UnderlyingSystemType { get; }
```

Summary

Returns the system-supplied type that represents the current type.

Property Value

The underlying system type for the `System.Type`.

Description

This property is read-only.

Behaviors

As described above.