

# 1 System.Reflection.EventInfo Class

```
2 [ILAsm]  
3 .class public abstract EventInfo extends System.Reflection.MemberInfo  
4 [C#]  
5 public abstract class EventInfo: MemberInfo
```

## 6 Assembly Info:

- 7 • *Name:* mscorlib
- 8 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- 9 • *Version:* 2.0.x.x
- 10 • *Attributes:*
  - 11 ○ CLSCompliantAttribute(true)

## 12 Summary

13 Provides access to event metadata.

## 14 Inherits From: System.Reflection.MemberInfo

15

16 **Library:** Reflection

17

18 **Thread Safety:** This type is safe for multithreaded operations.

19

## 20 Description

21 Events are handled by delegates. An event listener supplies an event-handler delegate  
22 that is invoked whenever the event is raised by an event source. In order to connect to  
23 the event source, the event listener adds this delegate to the invocation list of the  
24 source. When the event is raised, the event-handler delegate invokes the methods in its  
25 invocation list. The `System.Reflection.EventInfo.GetAddMethod`,  
26 `System.Reflection.EventInfo.AddEventHandler`,  
27 `System.Reflection.EventInfo.GetRemoveMethod`, and  
28 `System.Reflection.EventInfo.RemoveEventHandler` methods, and the delegate type  
29 of the event-handler associated with an event, are required to be marked in the  
30 metadata.

31

32 [*Note:* For information on delegates, see the `System.Delegate` class overview.]

33

34

35

36 [*Note:* For information on events, see Partitions I and II of the CLI specification.]

37

38

39

# 1 EventInfo() Constructor

```
2 [ILAsm]  
3 family rtspecialname specialname instance void .ctor()  
4 [C#]  
5 protected EventInfo()
```

## 6 Summary

7 Constructs a new instance of the `System.Reflection.EventInfo` class.

8

# 1 EventInfo.AddEventHandler(System.Object, 2 System.Delegate) Method

```
3 [ILAsm]  
4 .method public hidebysig instance void AddEventHandler(object target,  
5 class System.Delegate handler)  
  
6 [C#]  
7 public void AddEventHandler(object target, Delegate handler)
```

## 8 Summary

9 Adds the specified event handler delegate to the specified event source.

## 10 Parameters

Parameter	Description
<i>target</i>	An object that represents an event source.
<i>handler</i>	A System.Delegate instance to be added to <i>target</i> that references methods to be invoked when the event reflected by the current instance is raised by <i>target</i> .

11

## 12 Description

13 Each time the event reflected by the current instance is raised by *target*, the methods in  
14 the invocation list of *handler* are invoked.

## 15 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>handler</i> is not the same System.Type as the event handler delegate declared for the event reflected by the current instance.
<b>System.Reflection.TargetException</b>	The event reflected by the current instance is non-static, and <i>obj</i> is null or is of a type that does not implement the event reflected by the current instance.

16

17

# 1 EventInfo.GetAddMethod(System.Boolean) 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual abstract class  
5 System.Reflection.MethodInfo GetAddMethod(bool nonPublic)  
  
6 [C#]  
7 public abstract MethodInfo GetAddMethod(bool nonPublic)
```

## 8 Summary

9 Returns the method used to add an event handler delegate to an event source for the  
10 event reflected by the current instance, specifying whether or not to return non-public  
11 methods.

## 12 Parameters

Parameter	Description
<i>nonPublic</i>	A System.Boolean value that specifies whether non-public methods can be returned by this method. Specify true to return non-public methods; otherwise, specify false.

## 13 14 Return Value

15 A System.Reflection.MethodInfo instance that reflects the method used to add an  
16 event handler delegate to an event source for the event reflected by the current  
17 instance, if found; otherwise, returns null.

## 18 Description

19 [Note: The returned method is used to add an event-handler delegate to the invocation  
20 list of an event source. Typically, the method has the following signature format:

```
21 add_<EventName>( <EventHandlerType> handler )  
22  
23  
24 ]
```

## 25 Behaviors

26 As described above.

27

## 28 Exceptions

Exception	Condition
<b>System.MethodAccessException</b>	<i>nonPublic</i> is true, the method used to add an event handler delegate is non-public, and the caller does not have permission to reflect on non-public methods.

1

2 **Permissions**

Permission	Description
<b>System.Security.Permissions.ReflectionPermission</b>	Requires permission to reflect non-public members of a type in loaded assemblies. See <code>System.Security.Permissions.ReflectionPermissionFlag.TypeInformation</code> .

3

4

# 1 **MethodInfo.GetAddMethod()** Method

```
2 [ILAsm]  
3 .method public hidebysig instance class System.Reflection.MethodInfo  
4 GetAddMethod()  
  
5 [C#]  
6 public MethodInfo GetAddMethod()
```

## 7 **Summary**

8 Returns the public method used to add an event handler delegate to an event source for  
9 the event reflected by the current instance.

## 10 **Return Value**

11 A `System.Reflection.MethodInfo` instance that reflects the public method used to add  
12 an event handler delegate to an event source for the event reflected by the current  
13 instance, if found; otherwise, returns null.

## 14 **Description**

15 This method is equivalent to `System.Reflection.EventInfo.GetAddMethod(false)`.  
16

17 [*Note:* The returned method is used to add an event-handler delegate to the invocation  
18 list of an event source. Typically, the method has the following signature format:

```
19 add_<EventName>( <EventHandlerType> handler )  
20  
21 ]  
22
```

23

# 1 EventInfo.GetRaiseMethod(System.Boolean) 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual abstract class  
5 System.Reflection.MethodInfo GetRaiseMethod(bool nonPublic)  
  
6 [C#]  
7 public abstract MethodInfo GetRaiseMethod(bool nonPublic)
```

## 8 Summary

9 Returns the method that is called when the event reflected by the current instance is  
10 raised, specifying whether the method to be returned is public or non-public.

## 11 Parameters

Parameter	Description
<i>nonPublic</i>	A <code>System.Boolean</code> value that specifies whether non-public methods can be returned by this method. Specify <code>true</code> to return non-public methods; otherwise, specify <code>false</code> .

## 12 13 Return Value

14 A `System.Reflection.MethodInfo` instance that reflects the method that is called when  
15 the event reflected by the current instance is raised, if found; otherwise, returns `null`.

## 16 Behaviors

17 As described above.

## 18 19 Exceptions

Exception	Condition
<b>System.MethodAccessException</b>	<i>nonPublic</i> is <code>true</code> , the method used to raise the event is non-public, and the caller does not have permission to reflect on non-public methods.

## 20 21 Permissions

Permission	Description
<b>System.Security.Permissions.ReflectionPermission</b>	Requires permission to reflect non-public members of a type in loaded assemblies. See <code>System.Security.Permissions.ReflectionPermissionFlag.TypeInformation</code> .

1

2

# 1 **MethodInfo.GetRaiseMethod()** Method

```
2 [ILAsm]  
3 .method public hidebysig instance class System.Reflection.MethodInfo  
4 GetRaiseMethod()  
5 [C#]  
6 public MethodInfo GetRaiseMethod()
```

## 7 **Summary**

8 Returns the public method that is called when the event reflected by the current  
9 instance is raised.

## 10 **Return Value**

11 A `System.Reflection.MethodInfo` instance that reflects the public method that is  
12 called when the event reflected by the current instance is raised, if found; otherwise,  
13 returns `null`.

14

1  
2 **MethodInfo.GetRemoveMethod(System.Boolean)**  
3 **Method**

```
4 [ILAsm]  
5 .method public hidebysig virtual abstract class  
6 System.Reflection.MethodInfo GetRemoveMethod(bool nonPublic)  
7 [C#]  
8 public abstract MethodInfo GetRemoveMethod(bool nonPublic)
```

9 **Summary**

10 Returns the method used to remove an event-handler delegate from the event reflected  
11 by the current instance, specifying whether or not to return non-public methods.

12 **Parameters**

Parameter	Description
<i>nonPublic</i>	A System.Boolean value that specifies whether non-public methods can be returned by this method. Specify true to return non-public methods; otherwise, specify false.

13  
14 **Return Value**

15 A System.Reflection.MethodInfo instance that reflects the method used to remove an  
16 event handler delegate from the event reflected by the current instance, if found;  
17 otherwise, returns null.

18 **Description**

19 [Note: Typically, the method has the following signature format:  
20 remove\_<EventName>(<EventHandlerType> handler)  
21  
22 ]  
23

24 **Behaviors**

25 As described above.

26  
27 **Exceptions**

Exception	Condition
<b>System.MethodAccessException</b>	<i>nonPublic</i> is true, the method used to remove an event handler delegate is non-public, and the caller does not have permission to reflect on non-public methods.

1

2 **Permissions**

Permission	Description
<b>System.Security.Permissions.ReflectionPermission</b>	Requires permission to reflect non-public members of a type in loaded assemblies. See <code>System.Security.Permissions.ReflectionPermissionFlag.TypeInformation</code> .

3

4

# 1 **EventInfo.GetRemoveMethod()** Method

```
2 [ILAsm]  
3 .method public hidebysig instance class System.Reflection.MethodInfo  
4 GetRemoveMethod()  
  
5 [C#]  
6 public MethodInfo GetRemoveMethod()
```

## 7 **Summary**

8 Returns the public method used to remove an event-handler delegate from the event  
9 reflected by the current instance.

## 10 **Return Value**

11 A `System.Reflection.MethodInfo` instance that reflects the public method used to  
12 remove an event handler delegate from the event reflected by the current instance, if  
13 found; otherwise, returns null.

## 14 **Description**

15 This method is equivalent to `System.Reflection.EventInfo.GetRemoveMethod(false)`.

16  
17 [*Note:* Typically, the method has the following signature format:

```
18 remove_<EventName>(<EventHandlerType> handler)  
19  
20  
21 ]
```

22

1  
2 **EventInfo.RemoveEventHandler(System.Object, System.Delegate) Method**  
3

```
4 [ILAsm]  
5 .method public hidebysig instance void RemoveEventHandler(object target,  
6 class System.Delegate handler)  
  
7 [C#]  
8 public void RemoveEventHandler(object target, Delegate handler)
```

9 **Summary**

10 Removes the specified event handler delegate from the specified event source.

11 **Parameters**

Parameter	Description
<i>target</i>	An object that represents an event source.
<i>handler</i>	A <code>System.Delegate</code> instance to be disassociated from the events reflected by the current instance that are raised by <i>target</i> .

12  
13 **Description**

14 After this method is invoked, subsequent events reflected by the current instance that  
15 are raised by *target* will no longer cause *handler* to invoke its methods.

16 **Exceptions**

Exception	Condition
<b>System.ArgumentException</b>	<i>handler</i> is not the same type <code>System.Type</code> as the event handler delegate declared for the event reflected by the current instance.

17  
18

# 1 EventInfo.Attributes Property

```
2 [ILAsm]  
3 .property valuetype System.Reflection.EventAttributes Attributes { public  
4 hidebysig virtual abstract specialname valuetype  
5 System.Reflection.EventAttributes get_Attributes() }  
6 [C#]  
7 public abstract EventAttributes Attributes { get; }
```

## 8 Summary

9 Gets the attributes of the event reflected by the current instance.

## 10 Property Value

11 A System.Reflection.EventAttributes value that specifies the attributes in the  
12 metadata of the event reflected by the current instance.

13

# 1 EventInfo.EventHandlerType Property

```
2 [ILAsm]  
3 .property class System.Type EventHandlerType { public hideby sig  
4 specialname instance class System.Type get_EventHandlerType() }  
  
5 [C#]  
6 public Type EventHandlerType { get; }
```

## 7 Summary

8 Gets the `System.Type` of the event-handler `System.Delegate` associated with the event  
9 reflected by the current instance.

## 10 Property Value

11 A `System.Type` that represents the type of the event-handler `System.Delegate`  
12 associated with the event reflected by the current instance. Returns `null` if the method  
13 used to add a delegate to the event is not public and is in a loaded assembly, and the  
14 caller does not have the required permission.

## 15 Description

16 This property is read-only.

## 17 Permissions

Permission	Description
<b>System.Security.Permissions.ReflectionPermission</b>	Requires permission to reflect non-public members of a type in loaded assemblies. See <code>System.Security.Permissions.ReflectionPermissionFlag.TypeInformation</code> .

18  
19