

# System.Action<T1,T2,T3,T4> Delegate

```
[ILAsm]
.class public sealed System.Action`4<T1,T2,T3, T4> extends
System.MulticastDelegate

[C#]
public delegate void Action<in T1,in T2,in T3,in T4>(T1 arg1, T2 arg2, T3
arg3, T4 arg4);
```

## Assembly Info:

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

## Summary

Encapsulates a method that has four parameters and does not return a value.

## Parameters

Parameter	Description
<i>arg1</i>	The first parameter of the method that this delegate encapsulates.
<i>arg2</i>	The second parameter of the method that this delegate encapsulates.
<i>arg3</i>	The third parameter of the method that this delegate encapsulates.
<i>arg4</i>	The fourth parameter of the method that this delegate encapsulates.

## Inherits From: System.MulticastDelegate

**Library:** BCL

## Description

You can use the `System.Action`4<T1,T2,T3,T4>` delegate to pass a method as a parameter without explicitly declaring a custom delegate. The encapsulated method must correspond to the method signature that is defined by this delegate. This means that the encapsulated method must have four parameters that are all passed to it by value, and it must not return a value. Typically, such a method is used to perform an operation.

[Note: To reference a method that has four parameters and returns a value, use the generic `System.Func`5<T1,T2,T3,T4,TResult>` delegate instead.

]

When you use the `System.Action<T1,T2,T3,T4>` delegate, you do not have to explicitly define a delegate that encapsulates a method with four parameters.

You can also use the `System.Action`4<T1,T2,T3,T4>` delegate with anonymous methods in C#. (For an introduction to anonymous methods, see the C# standard.)