

# 1 System.Action<-T1,-T2,-T3> Delegate

```
2 [ILAsm]  
3 .class public sealed System.Action`3<-T1,-T2,-T3> extends  
4 System.MulticastDelegate  
  
5 [C#]  
6 public delegate void Action<in T1,in T2,in T3>(T1 arg1, T2 arg2, T3 arg3);
```

## 7 Assembly Info:

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

## 13 Summary

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

## 15 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.

16

## 17 Inherits From: System.Delegate

18

19 **Library:** BCL

20

## 21 Description

22 You can use the `System.Action`3<T1,T2,T3>` delegate to pass a method as a  
23 parameter without explicitly declaring a custom delegate. The encapsulated method  
24 must correspond to the method signature that is defined by this delegate. This means  
25 that the encapsulated method must have three parameters that are all passed to it by  
26 value, and it must not return a value. Typically, such a method is used to perform an  
27 operation.

28

29 [*Note:* To reference a method that has three parameters and returns a value, use the  
30 generic `System.Func`4<T1,T2,T3,TResult>` delegate instead.

31

1 ]

2

3 When you use the `System.Action`3<T1, T2, T3>` delegate, you do not have to explicitly  
4 define a delegate that encapsulates a method with three parameters.

5

6 You can also use the `System.Action`3<T1, T2, T3>` delegate with anonymous methods  
7 in C#. (For an introduction to anonymous methods, see the C# standard.)

8