1 System.Action<-T> Delegate

```
[ILAsm]
.class public sealed serializable System.Action`1<-T> extends
System.MulticastDelegate

[C#]
public delegate void Action<in T>(T obj);
```

Assembly Info:

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

Summary

Encapsulates a method that has a single parameter and does not return a value.

Parameters

Parameter	Description
obj	The parameter of the method that this delegate encapsulates.

Inherits From: System.MulticastDelegate

Library: BCL

Description

You can use the System.Action`1<T> 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 one parameter that is 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 one parameter and returns a value, use the
generic System.Func`2<T1,TResult> delegate instead.

]

When you use the System.Action`1<T> delegate, you do not have to explicitly define a
delegate that encapsulates a method with one parameter.

You can also use the System.Action`1<T> delegate with anonymous methods in C#.

1   (For an introduction to anonymous methods, see the C# standard.)
2
3   The `System.Collections.Generic.List`1<T>.ForEach` and
4   `System.Array.ForEach``1` methods each take an `System.Action`1<T>` delegate as a
5   parameter. The method encapsulated by the delegate allows you to perform an action
6   on each element in the array or list.

7