An object be able to communicate back to the entity that created it using the callback mechanism. In .net delegates are the preferred means of communicating back to the object using the callback mechanism.

Delegate is a type-safe object that points to a method or list of methods that can be invoked later.

So, delegates store 3 main type of information – 1. Address of the method to be invoked, 2. Parameters of the methods, and 3. Return type of the method.

When the C# compiler processes a delegate, it automatically generates a sealed class deriving from System.MulticastDelegate: System.Delegate

Compiler generated **BinaryOp** has three public: Invoke() – called behind the scenes synchronously.

BeginInvoke() and EndInvoke() provide the ability to call the current method synchronously on a separate thread of execution.

Declare delegate syntax –

*Public delegate int BinaryOp(int a, int b);*

Crux of the compiler generated BinaryOp class :

*Sealed class BinaryOp: System.MulticastDelegate{*

*Public int Invoke(int x, int y):*

*Public IAsycnResult BeginInvoke(int x, int y, AsyncCallback cb, object state);*

*Public int EndInvoke(IAsycnResult result);*

*}*

**In-case of a ref or out parameter, they will be present in the EndInovke method too, as a parameter**.

e.g.:

*public sealed class MyOtherDelegate : System.MulticastDelegate*

*{*

*public string Invoke(out bool a, ref bool b, int c);*

*public IAsyncResult BeginInvoke(out bool a, ref bool b, int c,*

*AsyncCallback cb, object state);*

*public string EndInvoke(out bool a, ref bool b, IAsyncResult result);*

*}*

Core members to all the delegate type –

* Method- This property returns a System.Reflection.MethodInfo object that represent the detail of a static method maintained by a delegate.
* Target - if the method to be called is of an object (not static), then Target returns an object that represents the method maintained by the delegate, if the value returned by the target is null, then the method is static.
* Combine() – Adds a method to the list of the methods in delegate.
* GetInvocationList() – returns an array of the System.Delegate objects each representing a particular method that may be invoked.
* Remove() – Remove method
* RemoveAll()

Multicasting – the delgates can maintain a list of methods to point to, this is multicasting.

If multiple methods are registered in the delegate, then all will be called when the handler is called in the called object.

**How is delegates a callback mechanism??**

**A – you can call a method on the caller instance from the called object**

**How?**

**A – Register the method in the delegate object, and create a instance of this delegate in the called object**

Anonymous Methods –

* An anonymous method cannot access the *ref or out* parameter of the defining method.

**KeyWords – type-safe**