public sealed class BinaryOp : System.MulticastDelegate

{

...

// Used to invoke a method asynchronously.

public IAsyncResult BeginInvoke(int x, int y,

AsyncCallback cb, object state);

// Used to fetch the return value

// of the invoked method.

public int EndInvoke(IAsyncResult result);

}

The System.IAsyncResult Interface

The BeginInvoke() method always returns an object implementing the IAsyncResult interface, while

EndInvoke() requires an IAsyncResult-compatible type as its sole parameter.

The IAsyncResultcompatible

object returned from BeginInvoke() is basically a coupling mechanism that allows the

calling thread to obtain the result of the asynchronous method invocation at a later time via

EndInvoke().

The IAsyncResult interface (defined in the System namespace) is defined as follows:

public interface IAsyncResult

{

object AsyncState { get; }

WaitHandle AsyncWaitHandle { get; }

bool CompletedSynchronously { get; }

bool IsCompleted { get; }

}

In the simplest case, you are able to avoid directly invoking these members. All you have to do is

cache the IAsyncResult-compatible object returned by BeginInvoke() and pass it to EndInvoke() when

you are ready to obtain the result of the method invocation. As you will see, you are able to invoke the

members of an IAsyncResult-compatible object when you want to become “more involved” with the

process of fetching the method’s return value.

**Note** If you asynchronously invoke a method that provides a void return value, you can simply “fire and forget.” In such cases, you will never need to cache the IAsyncResult-compatible object or call EndInvoke() in the first place (as there is no return value to retrieve).