An asynchronous operation that uses the [IAsyncResult](https://docs.microsoft.com/en-us/dotnet/api/system.iasyncresult) design pattern is implemented as two methods named BeginOperationName and EndOperationName that begin and end the asynchronous operation OperationName respectively. For example, the [FileStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream) class provides the [BeginRead](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream.beginread) and [EndRead](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream.endread) methods to asynchronously read bytes from a file. These methods implement the asynchronous version of the [Read](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream.read) method.

After calling BeginOperationName, an application can continue executing instructions on the calling thread while the asynchronous operation takes place on a different thread. For each call to BeginOperationName, the application should also call EndOperationName to get the results of the operation.

…..

A BeginOperationName method takes any parameters declared in the signature of the synchronous version of the method that are passed by value or by reference. Any out parameters are not part of the BeginOperationName method signature. The BeginOperationName method signature also includes two additional parameters. The first of these defines an [AsyncCallback](https://docs.microsoft.com/en-us/dotnet/api/system.asynccallback) delegate that references a method that is called when the asynchronous operation completes. The caller can specify null (Nothing in Visual Basic) if it does not want a method invoked when the operation completes. The second additional parameter is a user-defined object. This object can be used to pass application-specific state information to the method invoked when the asynchronous operation completes(*SyncCallback delegate*). If a BeginOperationName method takes additional operation-specific parameters, such as a byte array to store bytes read from a file, the [AsyncCallback](https://docs.microsoft.com/en-us/dotnet/api/system.asynccallback) and application state object are the last parameters in the BeginOperationName method signature.

BeginOperationName returns control to the calling thread immediately. If the BeginOperationName method throws exceptions, the exceptions are thrown before the asynchronous operation is started. If the BeginOperationName method throws exceptions, the callback method is not invoked.

## **Ending an Asynchronous Operation**

The EndOperationName method ends asynchronous operation OperationName. The return value of the EndOperationName method is the same type returned by its synchronous counterpart and is specific to the asynchronous operation. For example, the [EndRead](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream.endread) method returns the number of bytes read from a [FileStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream) and the [EndGetHostByName](https://docs.microsoft.com/en-us/dotnet/api/system.net.dns.endgethostbyname) method returns an [IPHostEntry](https://docs.microsoft.com/en-us/dotnet/api/system.net.iphostentry) object that contains information about a host computer. The EndOperationName method takes any out or ref parameters declared in the signature of the synchronous version of the method. In addition to the parameters from the synchronous method, the EndOperationName method also includes an [IAsyncResult](https://docs.microsoft.com/en-us/dotnet/api/system.iasyncresult) parameter. Callers must pass the instance(IAsyncResult instance) returned by the corresponding call to BeginOperationName.

If the asynchronous operation represented by the [IAsyncResult](https://docs.microsoft.com/en-us/dotnet/api/system.iasyncresult) object has not completed when EndOperationName is called, EndOperationName **blocks** the calling thread until the asynchronous operation is complete. Exceptions thrown by the asynchronous operation are thrown from the EndOperationName method. The effect of calling the EndOperationName method multiple times with the same [IAsyncResult](https://docs.microsoft.com/en-us/dotnet/api/system.iasyncresult) is not defined. Likewise, calling the EndOperationName method with an [IAsyncResult](https://docs.microsoft.com/en-us/dotnet/api/system.iasyncresult) that was not returned by the related Begin method is also not defined.

Application developers have several design choices for accessing the results of the asynchronous operation. The correct choice depends on whether the application has instructions that can execute while the operation completes. If an application cannot perform any additional work until it receives the results of the asynchronous operation, the application must block until the results are available. To block until an asynchronous operation completes, you can use one of the following approaches:

Call EndOperationName from the application’s main thread, blocking application execution until the operation is complete. For an example that illustrates this technique, see [Blocking Application Execution by Ending an Async Operation](https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/blocking-application-execution-by-ending-an-async-operation).

Use the [AsyncWaitHandle](https://docs.microsoft.com/en-us/dotnet/api/system.iasyncresult.asyncwaithandle) to block application execution until one or more operations are complete. For an example that illustrates this technique, see [Blocking Application Execution Using an AsyncWaitHandle](https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/blocking-application-execution-using-an-asyncwaithandle).

Applications that do not need to block while the asynchronous operation completes can use one of the following approaches:

* Poll for operation completion status by checking the [IsCompleted](https://docs.microsoft.com/en-us/dotnet/api/system.iasyncresult.iscompleted) property periodically and calling EndOperationName when the operation is complete. For an example that illustrates this technique, see [Polling for the Status of an Asynchronous Operation](https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/polling-for-the-status-of-an-asynchronous-operation).
* Use an [AsyncCallback](https://docs.microsoft.com/en-us/dotnet/api/system.asynccallback) delegate to specify a method to be invoked when the operation is complete. For an example that illustrates this technique, see [Using an AsyncCallback Delegate to End an Asynchronous Operation](https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/using-an-asynccallback-delegate-to-end-an-asynchronous-operation).

<https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/asynchronous-programming-model-apm>