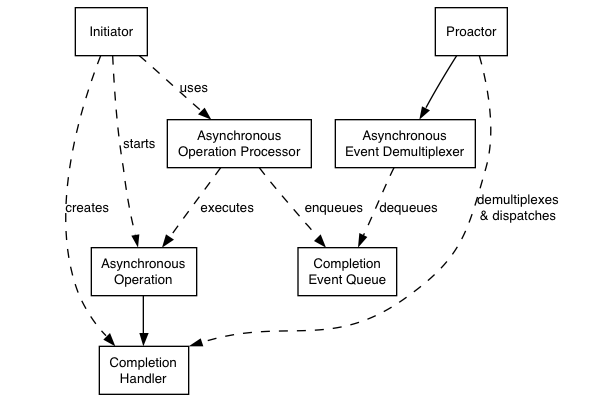
Proactor组件协作图



一、Proactor模式的参与者

Asynchronous Operation（异步操作）

一般是由应用程序投递的一项I/O请求，该I/O请求可以是网络连接的读取/写入请求，也可以是设备的I/O请求（比如硬盘的读取/写入），这个I/O请求是以异步方式执行的，也就是说，应用程序在投递一项I/O请求后不必阻塞的等待请求的完成，该异步请求会交给操作系统内部进行后续处理，并且在处理完成之后会以一定的方式通知应用程序：您发起的I/O请求已经完成了！

Asynchronous Operation Processor（异步操作的处理器）

支持异步机制的操作系统。比如Windows、Unix/Linux都支持对异步请求的处理。

Completion Handle（完成事件处理器）

应用程序设定的与异步请求相关的处理器对象（可以是函数或类）。当应用程序投递一个异步I/O请求时，必须指定一个与此请求相关联的完成事件处理器。当这个异步I/O请求处理完成后，操作系统会以一定的方式通知请求的投递者。此时应用程序当初设定的这个完成事件处理器会被Proactor框架回调。

Completion Event Handle（完成事件队列）

当异步请求处理器处理完一个异步I/O请求后，会产生一个完成事件并插入到完成事件队列中。在Windows的IOCP框架中，异步请求处理器即是操作系统内核，完成事件即是I/O完成包，完成事件队列即是由IOCP管理的完成队列。

Asynchronous Event Demultiplexer（完成事件分派器）

把完成事件队列中的完成事件分派给与其对应的完成事件处理器。

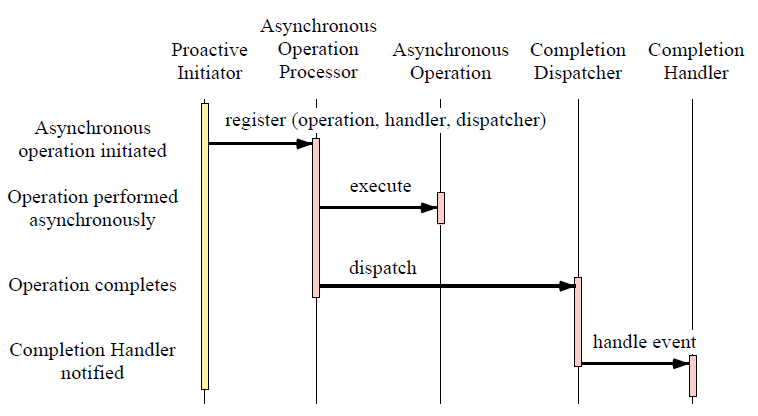
Initiator（初始化器）

投递异步I/O请求，并指定其完成事件处理器，使用异步操作处理器来处理请求。

Proactor框架对象

使用异步事件分派器从完成事件队列中取出完成事件，并分派给相应的事件处理器。

二、协作



三、实现

1、实现异步操作处理器

异步操作处理器主要由两个部分组成：提供给应用程序调用的**异步请求API**和**异步操作引擎。**

|  |
| --- |
| 异步请求API |
| 异步操作引擎 |