# The Reactor Design Pattern

The reactor design pattern is an event handling pattern for handling service requests delivered concurrently to a service handler by one or more inputs. The service handler then demultiplexes the incoming requests and dispatches them synchronously to associated request handlers.

## Intent

The Reactor pattern handles service requests that are delivered concurrently to an app by one or more clients. Each service in an Application may consists of several methods and is represented by a separate event handler that is responsible for dispatching service-specific requests. Dispatching of event handlers is performed by initiation dispatcher which manages the registered event handlers. Demultiplexing of service requests is performed by synchronous event demultiplexer.

## Example

Diagram

Description automatically generatedTo illustrated the Reactor pattern consider an event driven server for distributed logging shown on Figure 1 below:

Figure 1

Client applications use the logging service to record information about their status in a distributed environment. This status information commonly includes error notifications, debugging traces, and performance reports. Logging records are sent to a central logging server, which can write the records to various output devices, such as a terminal, a printer, a file, or a network management database.

The Logging Server shown on Figure 1 handles logging records and connection requests sent by clients.

Logging records and connection requests can arrive concurrently on multiple *handles*. A handle identifies network communication resources managed within an OS.

The logging server communicates with clients using a connection-oriented protocol such as TCP. Clients that want to log data must first send a connection request to the server. The server waits for these connection requests using a *handle factory* that listens on an address known to clients. When a connection request arrives, the handle factory establishes a connection between the client and the server by creating a new handle that represents an endpoint of the connection. This handle is returned to the server, which then waits for client service requests to arrive on the handle. Once clients are connected, they can send logging records concurrently to the server. The server receives these records via the connected socket handles.