CFRunLoopSource

Overview

A CFRunLoopSource object is an abstraction of an input source that can be put into a run loop. Input sources typically generate asynchronous events, such as messages arriving on a network port or actions performed by the user.

An input source type normally defines an API for creating and operating on objects of the type, as if it were a separate entity from the run loop, then provides a function to create a CFRunLoopSource for an object. The run loop source can then be registered with the run loop and act as an intermediary between the run loop and the actual input source type object. Examples of input sources include CFMachPort, CFMessagePort, and CFSocket.

There are two categories of sources. Version 0 sources, so named because the `version` field of their context structure is 0, are managed manually by the application. When a source is ready to fire, some part of the application, perhaps code on a separate thread waiting for an event, must call `CFRunLoopSourceSignal(_:)` to tell the run loop that the source is ready to fire. The run loop source for CFSocket is currently implemented as a version 0 source.

Version 1 sources are managed by the run loop and kernel. These sources use Mach ports to signal when the sources are ready to fire. A source is automatically signaled by the kernel when a message arrives on the source's Mach port. The contents of the message are given to the source to process when the source is fired. The run loop sources for CFMachPort and CFMessagePort are currently implemented as version 1 sources.

When creating your own custom run loop source, you can choose which version works best for you.

A run loop source can be registered in multiple run loops and run loop modes at the same time. When the source is signaled, whichever run loop that happens to detect the signal first will fire the source. Adding a source to multiple threads' run loops can be used to manage a pool of "worker" threads that is processing discrete sets of data, such as client-server messages over a network or entries in a job queue filled by a "manager" thread. As messages arrive or jobs get added to the queue, the source gets signaled and a random thread receives and processes the request.

Symbols

CFRunLoopSource Miscellaneous Functions

`func CFRunLoopSourceCreate(CFAllocator!, CFIndex, UnsafeMutablePointer<CFRunLoopSourceContext>!)`

Creates a CFRunLoopSource object.

[illegible]

Returns the context information for a CFRunLoopSource object.

`func CFRunLoopSourceGetOrder(CFRunLoopSource!)`

Returns the ordering parameter for a CFRunLoopSource object.

`func CFRunLoopSourceGetTypeID()`

Returns the type identifier of the CFRunLoopSource opaque type.

`func CFRunLoopSourceInvalidate(CFRunLoopSource!)`

Invalidates a CFRunLoopSource object, stopping it from ever firing again.

`func CFRunLoopSourceIsValid(CFRunLoopSource!)`

Returns a Boolean value that indicates whether a CFRunLoopSource object is valid and able to fire.

`func CFRunLoopSourceSignal(CFRunLoopSource!)`

Signals a CFRunLoopSource object, marking it as ready to fire.

Data Types

`CFRunLoopSourceContext`

A structure that contains program-defined data and callbacks with which you can configure a version 0 CFRunLoopSource's behavior.

`CFRunLoopSourceContext1`

A structure that contains program-defined data and callbacks with which you can configure a version 1 CFRunLoopSource's behavior.

`CFRunLoopSource`

A reference to a run loop source object.