

---

# Starscream



Starscream is a conforming WebSocket (RFC 6455) library in Swift.

## Features

- Conforms to all of the base Autobahn test suite.
- Nonblocking. Everything happens in the background, thanks to GCD.
- TLS/WSS support.
- Compression Extensions support (RFC 7692)

## Import the framework

First thing is to import the framework. See the Installation instructions on how to add the framework to your project.

```
1 import Starscream
```

## Connect to the WebSocket Server

Once imported, you can open a connection to your WebSocket server. Note that `socket` is probably best as a property, so it doesn't get deallocated right after being setup.

---

```
1 var request = URLRequest(url: URL(string: "http://localhost:8080")!)
2 request.timeoutInterval = 5
3 socket = WebSocket(request: request)
4 socket.delegate = self
5 socket.connect()
```

After you are connected, there is either a delegate or closure you can use for process WebSocket events.

### Receiving data from a WebSocket

`didReceive` receives all the WebSocket events in a single easy to handle enum.

```
1 func didReceive(event: WebSocketEvent, client: WebSocket) {
2     switch event {
3     case .connected(let headers):
4         isConnected = true
5         print("websocket is connected: \(headers)")
6     case .disconnected(let reason, let code):
7         isConnected = false
8         print("websocket is disconnected: \(reason) with code: \(code)"
9             )
10    case .text(let string):
11        print("Received text: \(string)")
12    case .binary(let data):
13        print("Received data: \(data.count)")
14    case .ping(_):
15        break
16    case .pong(_):
17        break
18    case .viabilityChanged(_):
19        break
20    case .reconnectSuggested(_):
21        break
22    case .cancelled:
23        isConnected = false
24    case .error(let error):
25        isConnected = false
26        handleError(error)
27        case .peerClosed:
28            break
29    }
```

The closure of this would be:

```
1 socket.onEvent = { event in
2     switch event {
```

---

```
3         // handle events just like above...
4     }
5 }
```

## Writing to a WebSocket

### write a binary frame

The `writeData` method gives you a simple way to send `Data` (binary) data to the server.

```
1 socket.write(data: data) //write some Data over the socket!
```

### write a string frame

The `writeString` method is the same as `writeData`, but sends text/string.

```
1 socket.write(string: "Hi Server!") //example on how to write text over
  the socket!
```

### write a ping frame

The `writePing` method is the same as `write`, but sends a ping control frame.

```
1 socket.write(ping: Data()) //example on how to write a ping control
  frame over the socket!
```

### write a pong frame

the `writePong` method is the same as `writePing`, but sends a pong control frame.

```
1 socket.write(pong: Data()) //example on how to write a pong control
  frame over the socket!
```

Starscream will automatically respond to incoming `ping` control frames so you do not need to manually send `pongs`.

However if for some reason you need to control this process you can turn off the automatic `ping` response by disabling `respondToPingWithPong`.

```
1 socket.respondToPingWithPong = false //Do not automatically respond to
  incoming pings with pongs.
```

In most cases you will not need to do this.

---

## disconnect

The disconnect method does what you would expect and closes the socket.

```
1 socket.disconnect()
```

The disconnect method can also send a custom close code if desired.

```
1 socket.disconnect(closeCode: CloseCode.normal.rawValue)
```

## Custom Headers, Protocols and Timeout

You can override the default websocket headers, add your own custom ones and set a timeout:

```
1 var request = URLRequest(url: URL(string: "ws://localhost:8080/")!)
2 request.timeoutInterval = 5 // Sets the timeout for the connection
3 request.setValue("someother protocols", forHTTPHeaderField: "Sec-
  WebSocket-Protocol")
4 request.setValue("14", forHTTPHeaderField: "Sec-WebSocket-Version")
5 request.setValue("chat,superchat", forHTTPHeaderField: "Sec-WebSocket-
  Protocol")
6 request.setValue("Everything is Awesome!", forHTTPHeaderField: "My-
  Awesome-Header")
7 let socket = WebSocket(request: request)
```

## SSL Pinning

SSL Pinning is also supported in Starscream.

Allow Self-signed certificates:

```
1 var request = URLRequest(url: URL(string: "ws://localhost:8080/")!)
2 let pinner = FoundationSecurity(allowSelfSigned: true) // don't
  validate SSL certificates
3 let socket = WebSocket(request: request, certPinner: pinner)
```

TODO: Update docs on how to load certificates and public keys into an app bundle, use the builtin pinner and TrustKit.

## Compression Extensions

Compression Extensions (RFC 7692) is supported in Starscream. Compression is enabled by default, however compression will only be used if it is supported by the server as well. You may enable compression by adding a `compressionHandler`:

---

```
1 var request = URLRequest(url: URL(string: "ws://localhost:8080/")!)
2 let compression = WSCompression()
3 let socket = WebSocket(request: request, completionHandler:
    compression)
```

Compression should be disabled if your application is transmitting already-compressed, random, or other uncompressable data.

## Custom Queue

A custom queue can be specified when delegate methods are called. By default `DispatchQueue.main` is used, thus making all delegate methods calls run on the main thread. It is important to note that all WebSocket processing is done on a background thread, only the delegate method calls are changed when modifying the queue. The actual processing is always on a background thread and will not pause your app.

```
1 socket = WebSocket(url: URL(string: "ws://localhost:8080/")!, protocols
    : ["chat", "superchat"])
2 //create a custom queue
3 socket.callbackQueue = DispatchQueue(label: "com.vluxestarscream.myapp")
```

## Example Project

Check out the SimpleTest project in the examples directory to see how to setup a simple connection to a WebSocket server.

## Requirements

Starscream works with iOS 8/10.10 or above for CocoaPods/framework support. To use Starscream with a project targeting iOS 7, you must include all Swift files directly in your project.

## Installation

### Swift Package Manager

The Swift Package Manager is a tool for automating the distribution of Swift code and is integrated into the `swift` compiler.

---

Once you have your Swift package set up, adding Starscream as a dependency is as easy as adding it to the `dependencies` value of your `Package.swift`.

```
1 dependencies: [  
2     .package(url: "https://github.com/daltoniam/Starscream.git", from:  
3         "4.0.6")  
4 ]
```

## CocoaPods

Check out Get Started tab on [cocoapods.org](https://cocoapods.org).

To use Starscream in your project add the following 'Podfile' to your project

```
1 source 'https://github.com/CocoaPods/Specs.git'  
2 platform :ios, '12.0'  
3 use_frameworks!  
4  
5 pod 'Starscream', '~> 4.0.6'
```

Then run:

```
1 pod install
```

## Carthage

Check out the Carthage docs on how to add a install. The [Starscream](#) framework is already setup with shared schemes.

Carthage Install

You can install Carthage with Homebrew using the following command:

```
1 $ brew update  
2 $ brew install carthage
```

To integrate Starscream into your Xcode project using Carthage, specify it in your `Cartfile`:

```
1 github "daltoniam/Starscream" >= 4.0.6
```

## Other

Simply grab the framework (either via git submodule or another package manager).

---

Add the [Starscream.xcodeproj](#) to your Xcode project. Once that is complete, in your “Build Phases” add the [Starscream.framework](#) to your “Link Binary with Libraries” phase.

### Add Copy Frameworks Phase

If you are running this in an OSX app or on a physical iOS device you will need to make sure you add the [Starscream.framework](#) to be included in your app bundle. To do this, in Xcode, navigate to the target configuration window by clicking on the blue project icon, and selecting the application target under the “Targets” heading in the sidebar. In the tab bar at the top of that window, open the “Build Phases” panel. Expand the “Link Binary with Libraries” group, and add [Starscream.framework](#). Click on the + button at the top left of the panel and select “New Copy Files Phase”. Rename this new phase to “Copy Frameworks”, set the “Destination” to “Frameworks”, and add [Starscream.framework](#) respectively.

### TODOs

- ☐ Proxy support
- ☐ Thread safe implementation
- ☐ Better testing/CI
- ☐ SSL Pinning/client auth examples

### License

Starscream is licensed under the Apache v2 License.

### Contact

#### Dalton Cherry

- <https://github.com/daltoniam>
- <http://twitter.com/daltoniam>
- <http://daltoniam.com>

#### Austin Cherry

- <https://github.com/acmacalister>
- <http://twitter.com/acmacalister>

- 
- <http://austincherry.me>