# QSocket: Interfaces

This thread has been locked by a moderator.

38

**IMPORTANT** If you haven't yet read Calling BSD Sockets from Swift, do that first.

Sometimes you need to get information about the local network interfaces on the device. All Apple platforms support the `getifaddrs` routine, which returns the list of network interfaces and their addresses. However, it works in terms of `struct sockaddr` values, which are hard to use from Swift. Here's an example of how you might use the `QSockAddr` primitives to return strings instead:

```
extension QSockAddr {

    /// Returns a list of interfaces that have an associated IPv4 or IPv6
    /// address.
    ///
    /// Equivalent to the `getifaddrs` BSD Sockets call.
    ///
    /// The list is in the same order as that returned by `getifaddrs`.
    /// ```

    public static func interfaceNamesAndAddresses() -> [(name: String, address: String)] {
        var addrList: UnsafeMutablePointer<ifaddrs>? = nil
        let err = getifaddrs(&addrList)
        // In theory we could check `errno` here but, honestly, what are gonna
        // do with that info?
        guard
            err >= 0,
            let first = addrList
        else { return [] }
        defer { freeifaddrs(addrList) }
        return sequence(first: first, next: { $0.pointee.ifa_next })
            .compactMap { addr in
                guard
                    let name = addr.pointee.ifa_name,
                    let sa = addr.pointee.ifa_addr,
                    [AF_INET, AF_INET6].contains(CInt(sa.pointee.sa_family)),
                    let (address, _) = try? QSockAddr.fromSockAddr(sa: sa, saLen: socklen_t(sa.pointee.sa_len))
                else { return nil }
                return (String(cString: name), address)
            }
    }
}
```

And once you have this primitive, you can add wrappers to get just the names, just the addresses, or the addresses grouped by the interface:

```
extension QSockAddr {

    public static func interfaceNames() -> [String] {
        interfaceNamesAndAddresses()
            .map { $0.name }
    }

    public static func interfaceAddresses() -> [String] {
        interfaceNamesAndAddresses()
            .map { $0.address }
    }

    public static func addressesByInterface() -> [String: [String]] {
        interfaceNamesAndAddresses()
            .reduce(into: [:]) { soFar, i in
                soFar[i.name, default: []].append(i.address)
            }
    }
}
```

**Note** If you're targeting macOS you have a lot more options in this space. Most notably, System Configuration framework has a dynamic store API that returns detailed information about the Mac's network state.

Share and Enjoy
—
Quinn "The Eskimo!" @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@" + "apple.com"`

Network

Reply

Posted 5 days ago by eskimo

Add a Comment