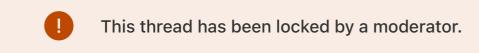


QSocket: DNS





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

DNS is pretty central to networking. When you connect to a service across the network, you usually start with a DNS name. In BSD Sockets you first resolve that name and then connect. This is one of the key problems with the API.

© 11

Apple APIs support connect-by-name semantics, which allows the networking stack to do a lot of work on your behalf. If you're using BSD Sockets in production code, you have to do all this work yourself. For more on this, see Connect by name, BSD Sockets best practices and DNS best practices in TN3151 Choosing the right networking API.

However, my test projects sometimes do need to resolve DNS names and so I've created a simple wrapper for this:

```
extension QSockAddr {
/// Resolves the specified DNS name and service to a list of IPv4 and IPv6
 /// addresses.
///
 /// Equivalent to the `getaddrinfo` BSD Sockets call.
 ///
/// The list may contain redundant values. For example, when you resolve
 /// `localhost` you get back 4 addresses, two for IPv4 and two for IPv6.
 ///
/// The list is in the same order as that returned by `getaddrinfo`.
 ///
 /// - Parameters:
 /// - host: The DNS name (or IP address) to resolve.
/// - service: A service name, like `"echo"`, or a port number, like `"4"`.
 public static func resolving(host: String, service: String) throws -> [(address: String, port: UInt16)] {
     var addrList: UnsafeMutablePointer<addrinfo>? = nil
     let err = getaddrinfo(host, service, nil, &addrList)
     guard err == 0 else { throw NetDBError(code: err) }
     defer { freeaddrinfo(addrList) }
     guard let first = addrList else { return [] }
     return try sequence(first: first, next: { $0.pointee.ai_next })
         .compactMap { addr in
             guard
                 [AF_INET, AF_INET6].contains(addr.pointee.ai_family),
                 let sa = addr.pointee.ai_addr,
                 case let saLen = addr.pointee.ai_addrlen,
                 saLen != 0
             else { return nil }
             return try QSockAddr.fromSockAddr(sa: sa, saLen: saLen)
// There's no wrapper for `getnameinfo` because /much/ more obscure than
// `getaddrinfo`. If you want to see an example of that, read the code for
 // `QSockAddr.fromSockAddr(sa:saLen:)`.
```

In my test projects I usually just pull the first item off the list and connect to that.

IMPORTANT If you use BSD Sockets in production code, connecting to the first address is not sufficient. For compatibility with a wide range of network environments, implement the Happy Eyeballs algorithm. See TN3151 Choosing the right networking API for the details.

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

This site contains user submitted content, comments and opinions and is for informational purposes only. Apple disclaims any and all liability for the acts, omissions and conduct of any third parties in connection with or related to your use of the site. All postings and use of the content on this site are subject to the Apple Developer Forums Participation Agreement.

