Debugging HTTP Server-Side Errors

This thread has been locked. Questions are automatically locked after two months of inactivity, or sooner if deemed necessary by a moderator.

1.7k

This post has been replaced by Debugging HTTP Server-Side Errors. I've left the original content in place below, but I recommend that you read the new article instead.

Apple's HTTP APIs can report an error in two different ways:

- A **transport error** is caused by a problem getting your request to, or getting the response from, the server. These are represented by an NSError, typically passed to your completion handler block or to a delegate method like

  `-URLSession:task:didCompleteWithError:`

  .
- An **server-side error** is caused by problems on the server itself. They are represented by the

  `statusCode`

  property of the NSHTTPURLResponse. You can interpret these errors using the information in Section 6 *Response Status Codes* of RFC 7231.

In some cases it's easy to interpret HTTP server-side errors. For example, a *404 Not Found* error means that the resource you've asked for does not exist. However, there are a variety of HTTP server-side errors where there's no way to determine, from the client side, what went wrong. These include all of the 5xx errors (like *500 Internal Server Error* and many of the 4xx erros (for example, with *400 Bad Request*, it's hard to know exactly why the server considers the request bad).

There are three ways to debug problems like this, as explained in the following sections.

HTTP Response Body

A lot of the time, when a server sends you a error response, it will include an HTTP body that explains what the problem is. You should look at the HTTP body to see if such an explanation is present. If it is, that's the easiest way to figure out what went wrong.

For example, consider a vanilla NSURLSession request like that shown below.

```
NSURLSession.sharedSession().dataTaskWithURL(NSURL(string: "http://www.apple.com/")!) { (responseBody, response,
error) in
    if let error = error {
        // handle transport error
    }
    let response = response as! NSHTTPURLResponse
    if response.statusCode / 100 != 2 {
        // handle HTTP server-side error
    }
    print("success")
}.resume()
```

If you get an HTTP server-side error then you'll find yourself at line 7. If you set a breakpoint at that line you can print

```
responseBody
```

to see if the server has any helpful hints as to what went wrong.

Compare Against a Working Client

If the response body does not contain any helpful hints as to what's causing the problem, another option is to compare your request to a request issued by a working client. For example, the server might accept a similar request from:

- a web browser, like Safari
- a command line tool, like

  `curl`

- an app running on a different platform

If you can find a working client then it's relatively straightforward to debug your problem:

1. take a packet trace of the request made by the working client
2. take a packet trace of the request made by your client
3. compare the two requests
4. fix any differences
5. retry with your fixed client
6. if things still fail, go back to step 2

There are some things to note here:

- For information on how to take a packet trace on Apple systems, see QA1176 Getting a Packet Trace.
- If you're using HTTPS, low-level packet traces are not helpful because Transport Layer Security (TLS), the *S* part of HTTPS, prevents you from seeing the HTTP request. You have a couple of options in that case:
  - If your server has a debugging mode that lets you see the plaintext request, look there.
  - If that's not an option, you can use a proxy-based HTTPS debugging tool, like Charles.
- Rather than try to make the requests exactly match the first time around, it's best to focus on the high-level stuff first. Does the URL path match? Does the HTTP method match? Does the

  `Content-Type`

  header match? What about the remaining headers? Does the request body match? If these all match and things still don't work, you might need to look at lower-level stuff, like the HTTP transfer encoding and, if you're using HTTPS, various TLS parameters.

Server-Side Debugging

If you don't have access to a working client, or you can't get things to work using the steps described in the previous section, your only remaining option is to debug this on the server. If you're lucky, the server will have documented debugging options that offer more insight into the failure. If not, you should escalate this via the support channel associated with your server software.

Share and Enjoy — Quinn "The Eskimo!" Apple Developer Relations, Developer Technical Support, Core OS/Hardware

```
let myEmail = "eskimo" + "1" + "@apple.com"
```

Network

Asked 5 years ago by eskimo

Reply to this question

