On FTP

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

704

Questions about FTP crop up from time-to-time here on DevForums. In most cases I write a general “don’t use FTP” response, but I don’t have time to go into all the details. I’ve created this post as a place to collect all of those details, so I can reference them in other threads.

IMPORTANT Apple’s official position on FTP is:

- Certain FTP APIs have been deprecated, and you should avoid using deprecated APIs.
- Apple has been slowly removing FTP support from the user-facing parts of our system. The most recent example of this is that we removed the

  `ftp`

  command-line tool in macOS 10.13.
- You should avoid the FTP protocol and look to adopt more modern alternatives.

The rest of this post is an informational explanation of the overall FTP picture.

Note This post is locked so I can keep it focused. If you have questions or comments, please do create a new thread in the Core OS > Networking topic area and I’ll respond there.

Don’t Use FTP

FTP is a very old and very crufty protocol. Certain things that seem obvious to us now — like being able to create a GUI client that reliably shows a directory listing in a platform-independent manner — are not possible to do in FTP. However, by far the biggest problem with FTP is that it provides *no* security. Specifically, the FTP protocol:

- Provides no on-the-wire privacy, so anyone can see the data you transfer
- Provides no client-authenticates-server authentication, so you have no idea whether you’re talking to the right server
- Provides no data integrity, allowing an attacker to munge your data in transit
- Transfers user names and passwords in the clear

Using FTP for anonymous downloads may be acceptable (see the note below) but most other uses of FTP are completely inappropriate for the modern Internet.

IMPORTANT You should only use FTP for anonymous downloads if you have an independent way to check the integrity of the data you’ve downloaded. For example, if you’re downloading a software update, you could use code signing to check its integrity. If you don’t check the integrity of the data you’ve downloaded, an attacker could substitute a malicious download instead. This would be especially bad in, say, the software update case.

The fundamental problems with the FTP protocol means that it is not a priority for Apple. This is reflected in the available APIs, which is the subject of the next section.

FTP APIs

Apple provides two FTP APIs:

- All Apple platforms provide FTP downloads via

  `NSURLSession`

- Most Apple platforms (everything except watchOS) support

  `CFFTPStream`

  , which allows for directory listings, downloads, uploads, and directory creation.

IMPORTANT

`CFFTPStream`

has been officially deprecated since 2016 (with the iOS 9 and macOS 10.11 SDKs). It still works about as well as it ever did, which is not particularly well. Specifically:

- There is at least one known crashing bug (r. 35745763), albeit one that occurs quite infrequently.
- There are clear implementation limitations — like the fact that

  `CFFTPCreateParsedResourceListing`

  assumes a MacRoman text encoding (r. 7420589) — that will not be fixed.

If you’re looking for an example of how to use these APIs, check out the SimpleFTPSample sample code.

Note This sample has not been updated since 2013 and is unlikely to ever be updated given Apple’s position on FTP.

Even though there is one non-deprecated API that supports FTP,

`NSURLSession`

, there are still major limitations in its use:

- `NSURLSession`

  only supports FTP downloads; there is no support for uploads or any other FTP operations
- `NSURLSession`

  does not support resumable FTP downloads [1]
- `NSURLSession`

  background sessions only support HTTP and HTTPS, so you can’t run FTP downloads in the background on iOS

If Apple’s FTP APIs are insufficient for your needs, you will need to write or acquire your own FTP library. Before you do that, however, you should consider switching to an alternative protocol. After all, if you’re going to go to the trouble of importing a large FTP library into your code base, you might as well import a library for a *better* protocol. The next section discusses some options in this space.

Alternative Protocols

There are numerous better alternatives to FTP:

- HTTPS is by far the best alternative to FTP, offering good security, good APIs on Apple platforms, good server support, and good network compatibility.
  Implementing traditional FTP operations over HTTPS can be a bit tricky. One possible way forward is to enable DAV extensions on the server.
- FTPS is FTP over TLS (aka SSL). While FTPS adds security to the protocol, which is very important, it still inherits many of FTP’s other problems. Personally I try to avoid this protocol.
- SFTP is a file transfer protocol that’s completely unrelated to FTP. It runs over SSH, making it a great alternative in many of the ad hoc setups that traditionally use FTP.

Apple does not have an API for either FTPS or SFTP, although on macOS you may be able to make some headway by invoking the

`sftp`

command-line tool.
Share and Enjoy — Quinn “The Eskimo!” Apple Developer Relations, Developer Technical Support, Core OS/Hardware

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

[1] Although you can implement resumable downloads using the lower-level

`CFFTPStream`

API, courtesy of the

`kCFStreamPropertyFTPFileTransferOffset`

property.
Change History

23 Feb 2018 — First posted.

Network

Asked 3 years ago by eskimo

Reply to this question

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.

Copyright © 2021 Apple Inc. All rights reserved. Terms of Use | Privacy Policy | License Agreements