ownCloud iOS App Manual

The ownCloud Team

Version: 11.9, June 09, 2022

Table of Contents

The Mobile App for iOS . . . 1
Getting the ownCloud iOS App . . . 2
Managing User Accounts . . . 3
- Introduction . . . 3
- Authentication Methods . . . 3
- Authentication Options . . . 3
- Manage Accounts . . . 4

Configure Settings . . . 8
- Introduction . . . 8
- Settings Screen . . . 8
- Security . . . 8
- Themes . . . 10
- Logging . . . 11
- Media Upload (Conversion) . . . 12

Managing Files . . . 13
- Introduction . . . 13
- Features . . . 13
- File Actions . . . 13
- Navigate Folders . . . 14
- Navigate Files . . . 15
- Sort Files and Folders . . . 15
- View Files . . . 16
- Folder Actions . . . 17
- Drag & Drop Files Between Apps (iPad-only) . . . 21

Offline Files and Folders . . . 22
- Introduction . . . 22
- Making a File or Folder Available Offline . . . 22
- Removing a File or Folder from Offline Storage . . . 22
- Viewing Offline Files . . . 23
- Storage . . . 25

Collaboration and Links . . . 26
- Introduction . . . 26
- Collaborate With Other Users on Files and Folders . . . 26
- Manage Links . . . 26

Quick Access . . . 30
- Introduction . . . 30
- Shares . . . 30
- Collection . . . 30

iOS App and iOS Files App Integration and 3rd Party Apps . . . 32
Introduction . . . 32
Work With Your ownCloud Data in iOS Files App . . . 32
Share From Other Apps (via iOS Files App) . . . 32
Task Scheduling . . . 33
Introduction . . . 33
Tasks . . . 33
Security . . . 34
Introduction . . . 34
Authentication . . . 34
Connections . . . 35
Data . . . 36
Miscellaneous . . . 36
iOS Frequently Asked Questions (FAQ) . . . 38
Introduction . . . 38
Usage . . . 38
Feature Requests . . . 38
Appendices . . . 40
Mobile Device Management (MDM) . . . 40
Troubleshooting . . . 62
Release Notes . . . 66

The Mobile App for iOS

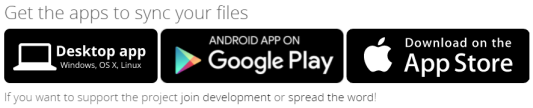
Accessing your files on your ownCloud server via the Web interface is easy and convenient. You can use any web browser on any operating system without installing special client software. However, the ownCloud iOS app offers several key advantages over the web interface; these include:

- A simplified interface that fits nicely on an iPhone or iPad
- Upload files easily from your device to ownCloud
- An Optional PIN for stronger security
- Share files with other ownCloud users
- Automatic synchronization of your files

Getting the ownCloud iOS App

To get the ownCloud iOS App, point Safari, or your favorite web browser, to your ownCloud server. Next, log in and navigate to **Settings › General (Personal)**. At the bottom of that page, you will see a link to the ownCloud app on iTunes.

Links to ownCloud's mobile apps.



You'll also find links and information on the ownCloud installation page.

Managing User Accounts

Introduction

With the iOS app, users can access their data from several ownCloud instances. They can use one or more accounts for the same server, or accounts for other ownCloud servers they have access to.

Authentication Methods

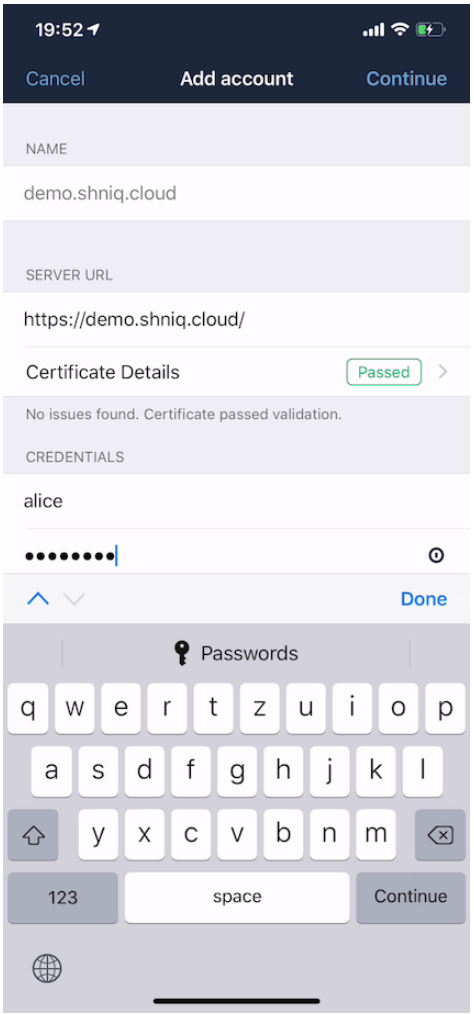
The iOS app supports two ways of authentication options for user accounts. These are:

- Basic Authentication
- OAuth 2.0

Authentication Options

Basic Authentication

- Login Credentials are stored on the device (secured in the iOS system keychain).
- Password manager support.



OAuth 2.0

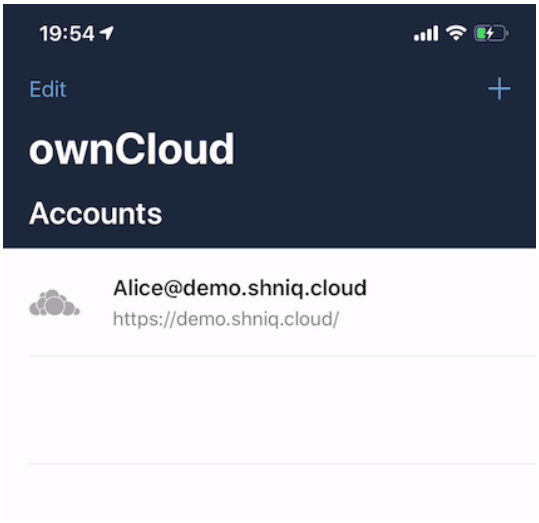
The OAuth2 implementation uses `SFAuthenticationSession`, which is described as a best practice by `RFC 8252` - when running under iOS 11. Under iOS 12, the OAuth2 implementation uses `ASWebAuthenticationSession`, which is the successor of `SFAuthenticationSession`. Benefits of using these APIs include:

- **Privilege separation:** Web content is run in a separate process.
- **Trustworthiness:** Apps can't inject code into or access the contents of the web view.
- **Convenience for the user:** Cookies from Safari are available to the web content inside the session.

Additionally, OAuth2 is the industry standard authorisation method, and the username and password are not stored on the device.

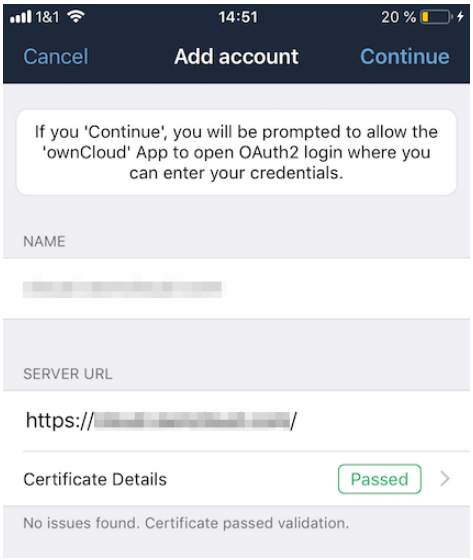
Manage Accounts

Add an Account

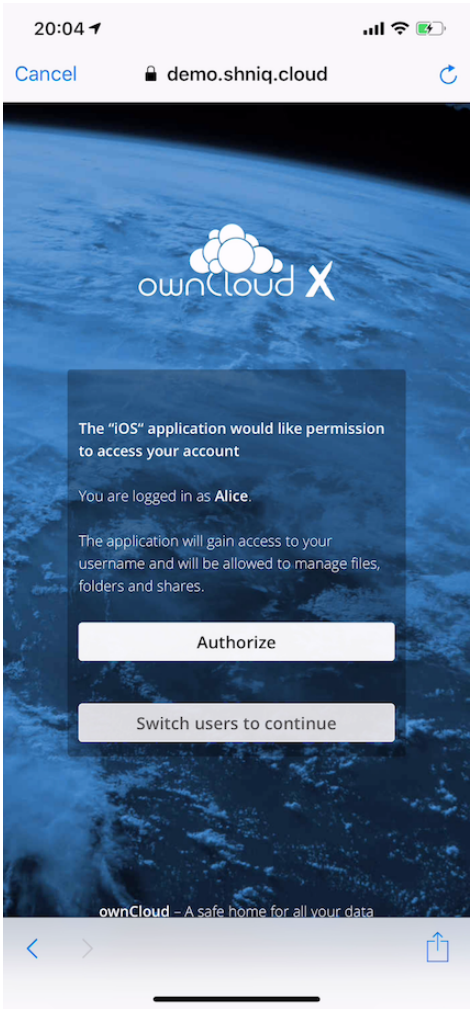


To add one or more user accounts to the iOS app, when in the "**Accounts**" view, click the plus (+) icon in the top right-hand corner. This opens the "**Add Account**" dialog, where you can enter the URL of the ownCloud server. After you enter it and click "Continue", the iOS app checks the authentication method and the validity of the SSL/TLS certificate (*if the server URL uses the HTTPS protocol*).

If the certificate is deemed to be valid, you will see a green "**Passed**" symbol near the bottom of the page, next to "**Certificate Details**", and the text "**No issues found. Certificate passed validation.**"



Click **[Continue]** and the app will prompt you if you want to use the supplied server URL to sign in to the app. You will then be redirected to the ownCloud server, where you can supply your username and password. After doing so, and submitting the form, you will then be asked if you want to give permission for the app to access your account.



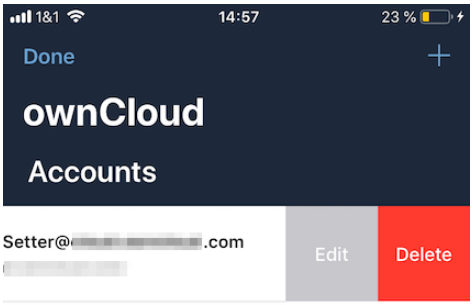
- If so, click **[Authorize]**.
- If not, click **[Cancel]**.
- If you clicked **[Authorize]**, you will then be returned to the Accounts screen, where you will

see your new account in the list.

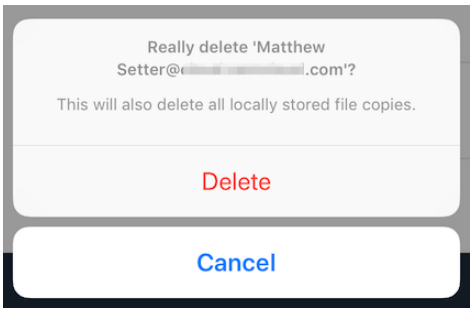
ownCloud server must have the OAuth2 app installed, configured, and enabled to use Two-Factor Authentication. Please contact your ownCloud administrator for more details.

Delete an Account

If you want to delete an account, when viewing the Accounts list, swipe left on the account that you want to delete and click **[Delete]**.



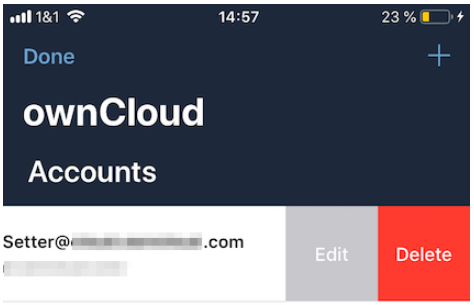
You will then be asked if you really want to delete that account.



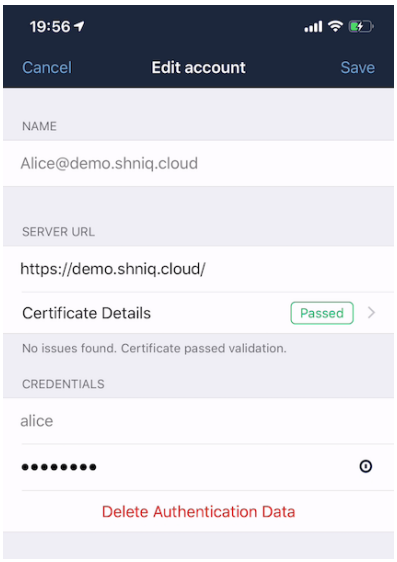
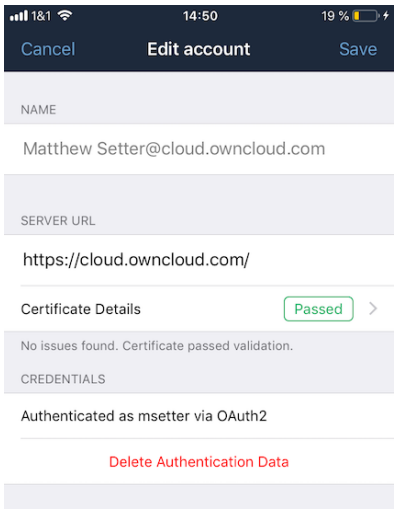
If you do, click **[Delete]**. Doing so deletes the account from the device, as well as all locally stored file copies. If you don't want to delete the account, click **[Cancel]**.

Edit Authentication

If you want to edit an account, when viewing the Accounts list, swipe left on the account that you want to edit and click **[Edit]**.



You will then be able to change the ownCloud server URL, and manage the authentication credentials. How the authentication credentials can be managed depends on the authentication type.

Basic Authentication	OAuth2 Authentication
The user is authenticated using Basic Authentication. In this setup, they will be able to enter a different password, as well as delete their authentication data.	The user is authenticated using OAuth2 authentication. In this setup, they will only be able to delete their OAuth2 authentication.
	

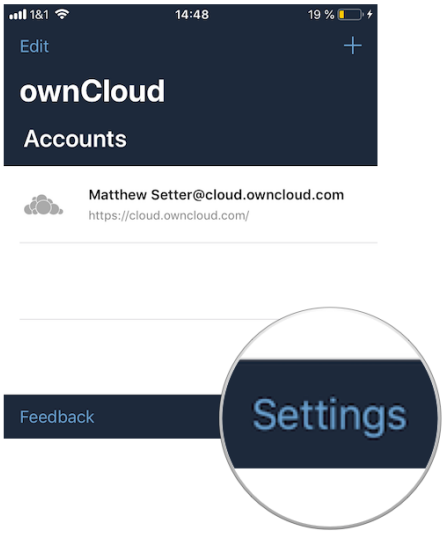
Configure Settings

Introduction

This document describes which settings are available and how to set them

Settings Screen

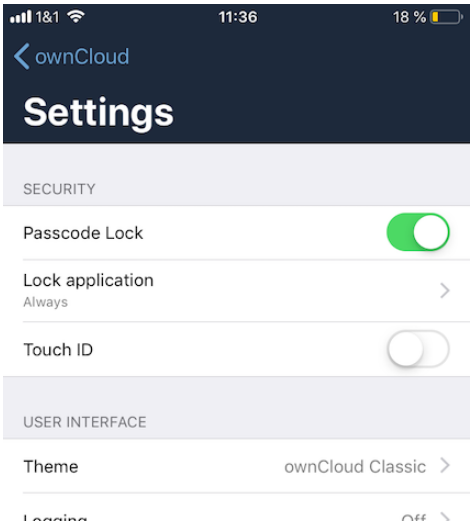
To manage the settings in ownCloud's iOS App for iPhone and iPad, click **[Settings]** in the bottom right-hand corner of the Accounts List view.



Security

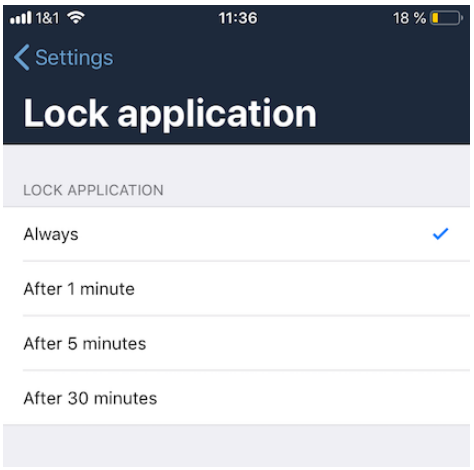
Passcode

To protect access to the iOS app with a Passcode, enable **[Passcode Lock]** in the Setting's Security section. You will then be prompted to enter, and repeat, a 4-digit Passcode. If a Passcode was set, the file provider extension in Files.app or in other third party apps will be locked too. The file provider presents a UI to unlock the Passcode.



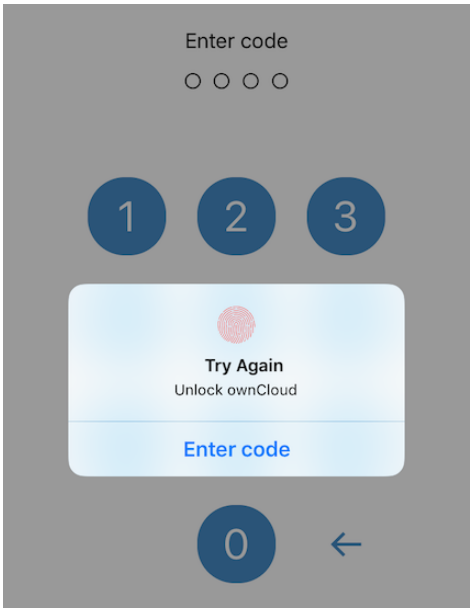
Lock Delay

When a Passcode is enabled, the app will be locked every time you change to another application. However, under **Settings › Security › Lock application**, you can choose to only lock the application after 1, 5, or 30 minutes, instead of "*immediately*", which is the default.



Biometrical Lock

After a Passcode has been created, a Biometrical Lock, or Touch ID, can also be used to gain access to the app. To enable it enable **[Touch ID]** in the Setting's *Security* section, and then enter your 4-digit Passcode. The next time you need to authorise access to the app, you will be able to enter either your Passcode, or use your stored biometrical data.



Trusted Certificates

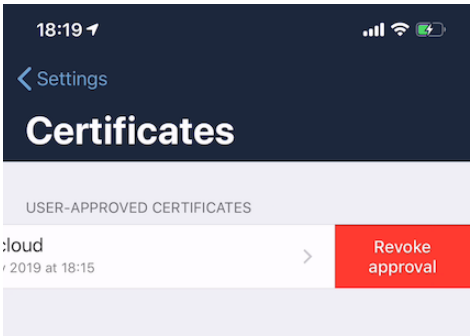
View Previously Approved Certificates

To view previously approved certificates, navigate to **Settings › Certificates** (for any one of your registered accounts), and you will see them listed in the "*User-Approved Certificates*" section.

Inspect Previously Approved Certificates Details

To view previously approved certificates, swipe left on any of the accounts in the accounts list and click **[Edit]**. Then, under "*SERVER URL*", click **[Certificate Details]**. You will then be able to see all of the certificate's details.

Revoke Previously Approved Certificates



To revoke one or more previously approved certificates, first navigate to **Settings › Certificates** (for any one of your registered accounts). Then, in the "*User-Approved Certificates*" section, swipe left on the certificate(s) that you wish to revoke and press **[Revoke approval]**.

Themes

The iOS app comes with three themes:

- Light

- Dark; and
- Classic

To change the theme, navigate to **Settings › Theme**, and pick the one that you want.

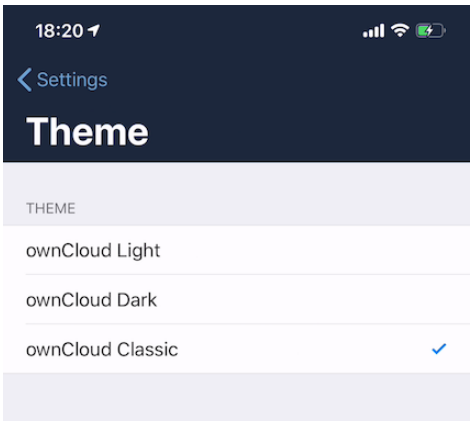
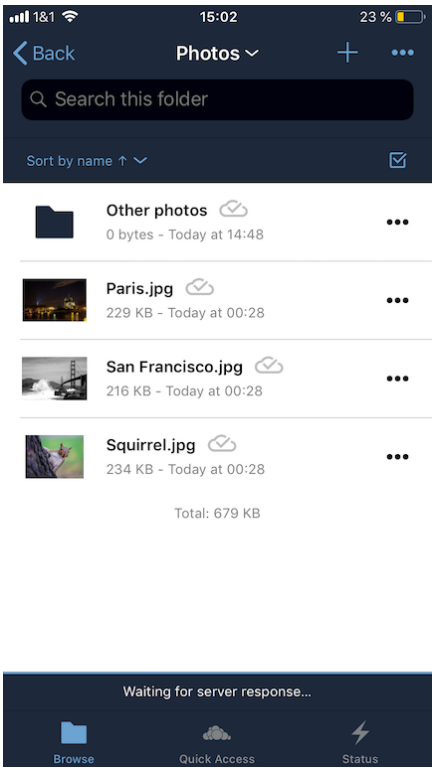
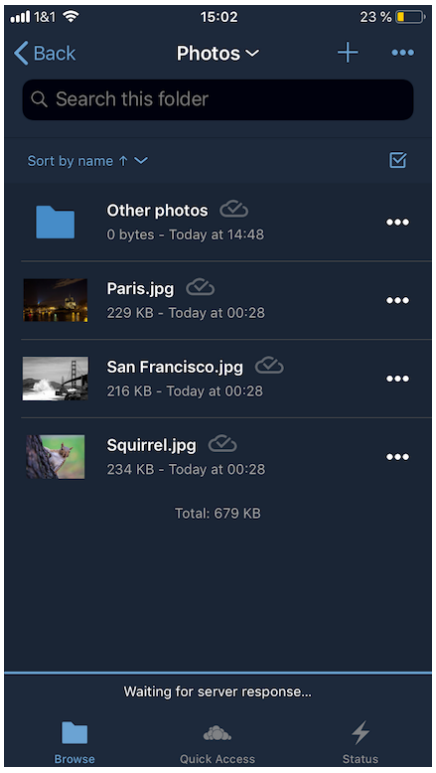
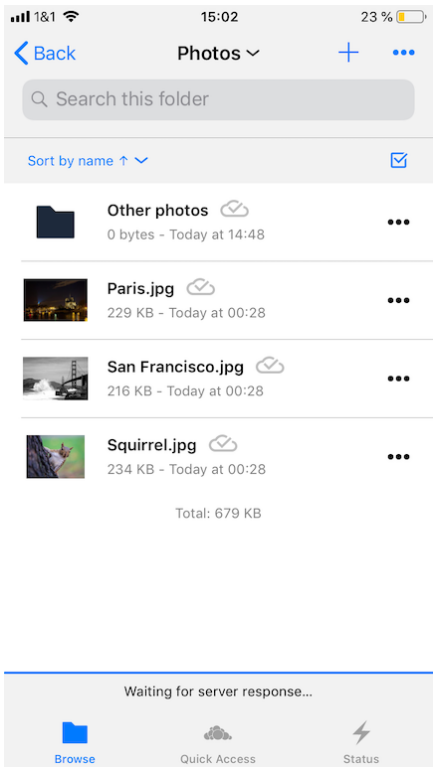


Table 1. The three themes in ownCloud's iOS App for iPhone and iPad.

Classic theme	Dark theme	Light theme
		

System Appearance (up from iOS 13)

When System Appearance is selected, ownCloud will automatically use the selected iOS system theme (light or dark) to reflect the system UI. Setting System Appearance is only available up from iOS 13.

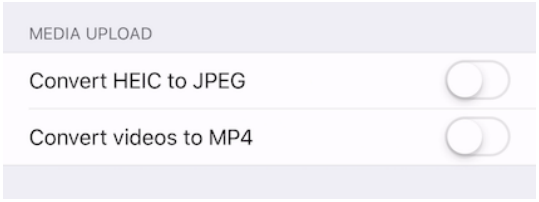
Logging

The ownCloud iOS app has built-in logging functionality, available under **Settings › Logging**. To

find out more, please refer to the logging section of the Troubleshooting guide.

Media Upload (Conversion)

When image and video files are uploaded, they can be converted to the industry-standard JPEG and MP4 respectively. This is not done by default.



Image

To convert (the very efficient) HEIC (High Efficiency Image File Format) images to more compatible JPEG images, enable **[Convert HEIC to JPEG]** under **Settings › Media Upload**.

Video

To convert the very efficient videos to more compatible MP4 videos, enable **[Convert videos to MP4]** under **Settings › Media Upload**.

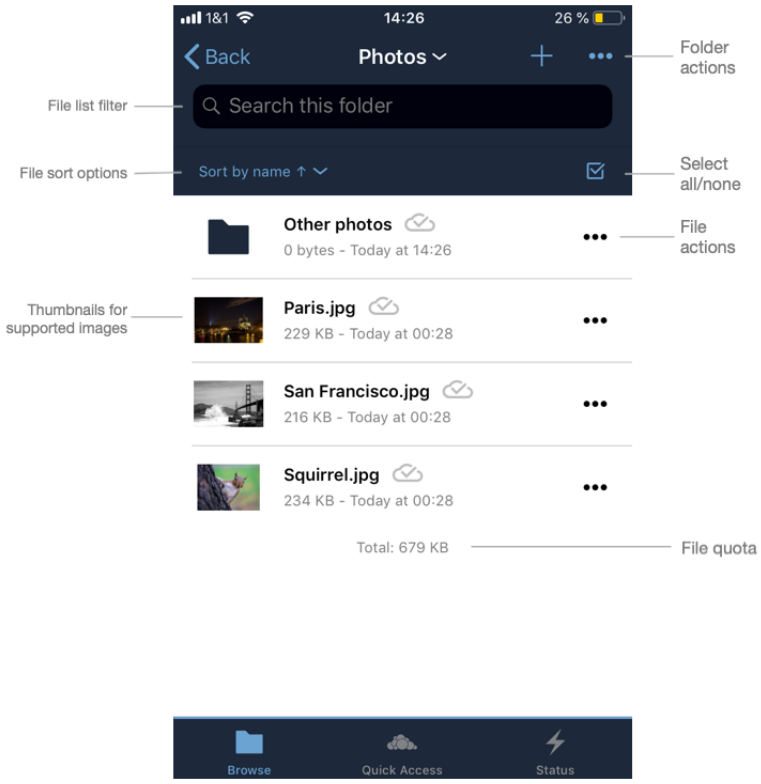
Managing Files

Introduction

The IOS app provides a number of features designed to make managing files as simple as possible.

Features

- File list filter
- File sort options
- Thumbnails for supported images
- File quota (only displayed in the root folder)
- File actions
- Folder actions
- Select all/none



Click on a file to view it and click on a folder to view its contents. To view further actions available for a file or folder, click the *More* icon on the far right-hand side.

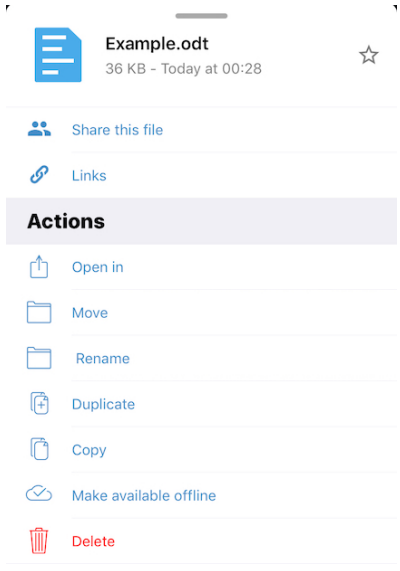
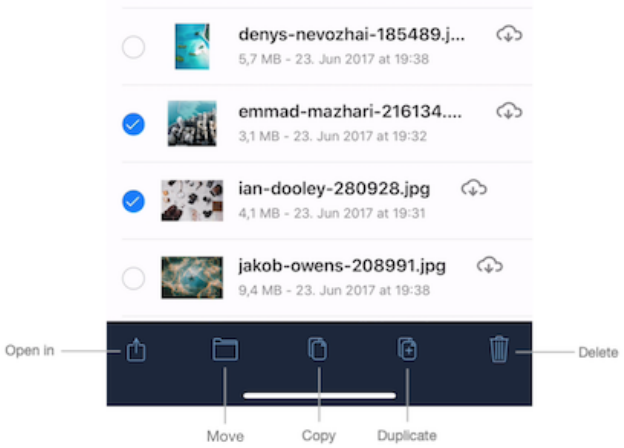
File Actions

The actions available for files are:

- Open in

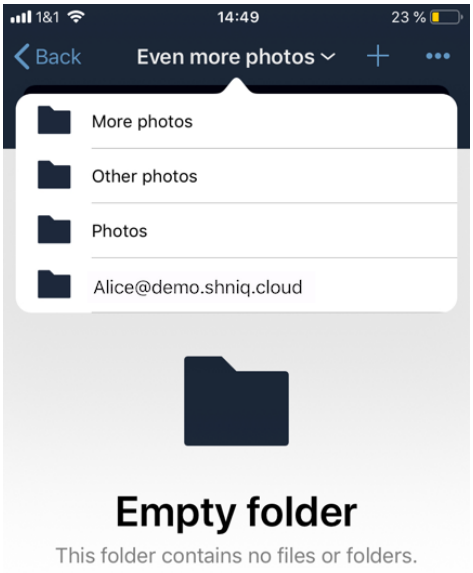
- Move
- Rename
- Duplicate
- Copy
- Delete
- Make Available Offline

However, file actions are handled slightly differently, depending on whether one or multiple files have been selected. You can see the differences in the two images below.

Individual File Actions	Multiple-Selected File Actions
File popup for actions for *individual* files	Move icon row for *multiple-selected* files
	

Navigate Folders

There are two ways to navigate folders (outside of the root folder). To go back to the parent folder, tap **[Back]** in the top left-hand corner. To navigate directly to **any** parent folder, tap on the current folder's name. When you do, you will see the names of all the parent folders, right up to the root folder, as in the image below.



Navigate Files

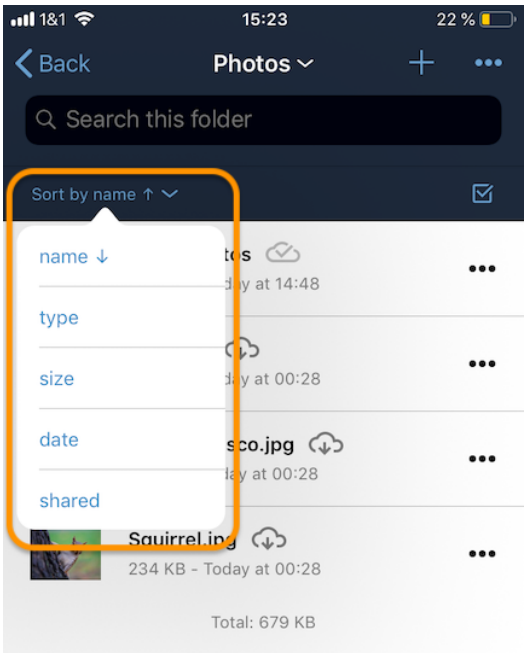
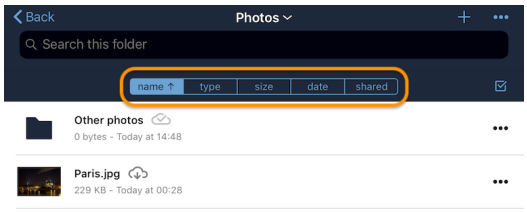
When navigating files, as you would expect, you can scroll up and down the files and folders list. In addition, you can also use the index bar, highlighted in the screenshot below, to speed up traversing through files and folders. As you slide your finger over each letter, you'll jump to the first file (or folder) that begins with that letter.

The index bar is only visible, if the sort order is `Name`.

Sort Files and Folders

By default, files and folders are sorted by name in ascending order, with folders sorted before files. However, files can be sorted in ascending and descending order by name, type, size, date, and shared. If you press the Sort by menu, you can change sort method and order. The first time you change the sort category, files and folders are sorted using that category in ascending order. If you

choose that category a second time, the sort order is inverted.

Sorting files and folders in portrait mode	Sorting files and folders in landscape mode
	

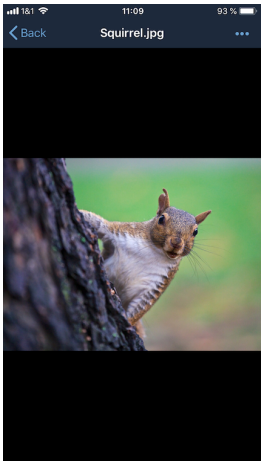
View Files

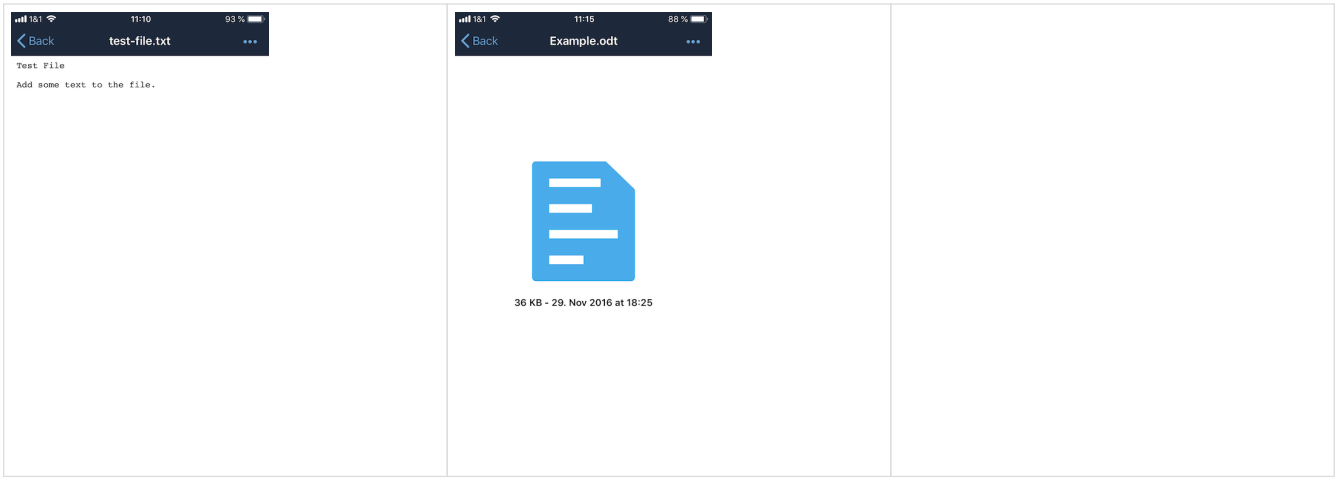
To view a file, tap on its name in the file list. Any file type supported by iOS Safari can be displayed in the app. Depending on the file type, the image will be able to viewed, or an icon for it, along with some file details, will be displayed.

If the file is not available locally on the device, you will see next to the file.

When you click on it, you will see next to it while it downloads.

Table 2. Viewing different file types

An image file	A video file	A PDF file
		
A text file	An ODT file.	



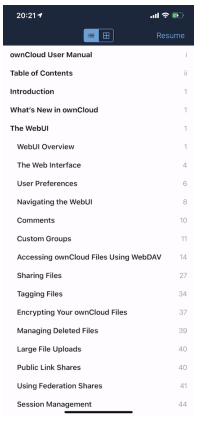
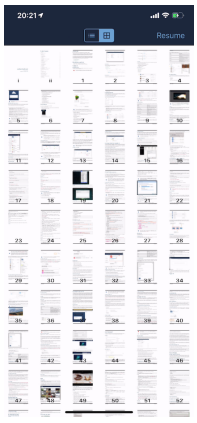
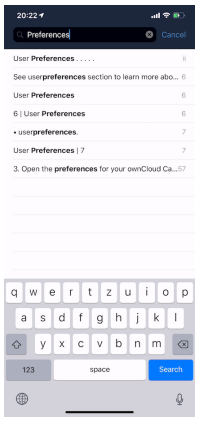
PDF Files

When viewing PDF files four UI options are available which make working with them easier; these are:

- A page selector
- Page thumbnails
- A Table of Contents
- File Search

You can see an example of each in the images below.

Table 3. PDF file functionality

A page selector	A table of contents	Page thumbnails	File search
			

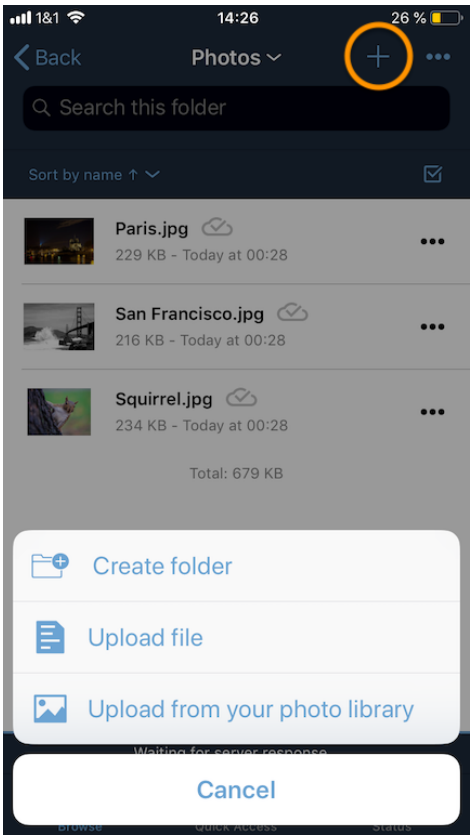
Video Files

Video files have the standard iOS video controls available, which include play, pause, AirPlay, volume, skip forward, skip back, close, and full screen.

Folder Actions

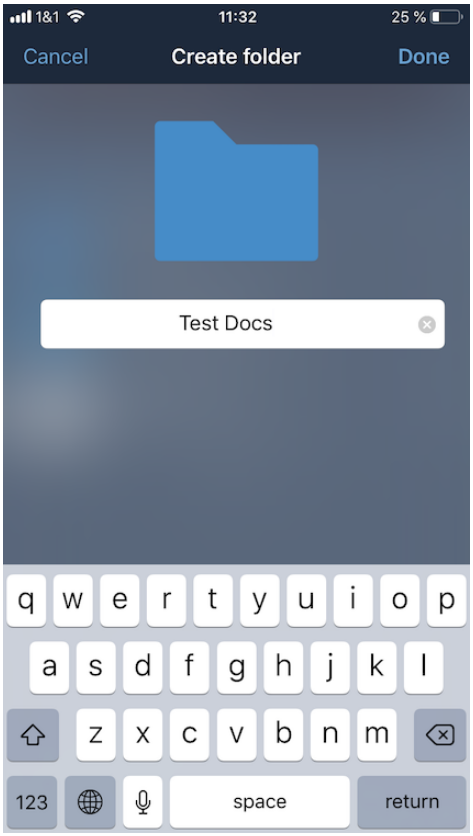
When working with folders, click the plus icon near the top right-hand corner, and three actions become available; these are:

- Create folder
- Upload files
- Upload file from your photo library
- Make available offline



Create Folder

To create a new folder, click **[Create folder]**, enter the name of the new folder, as in the image below, and click **[return]**.



Upload Files

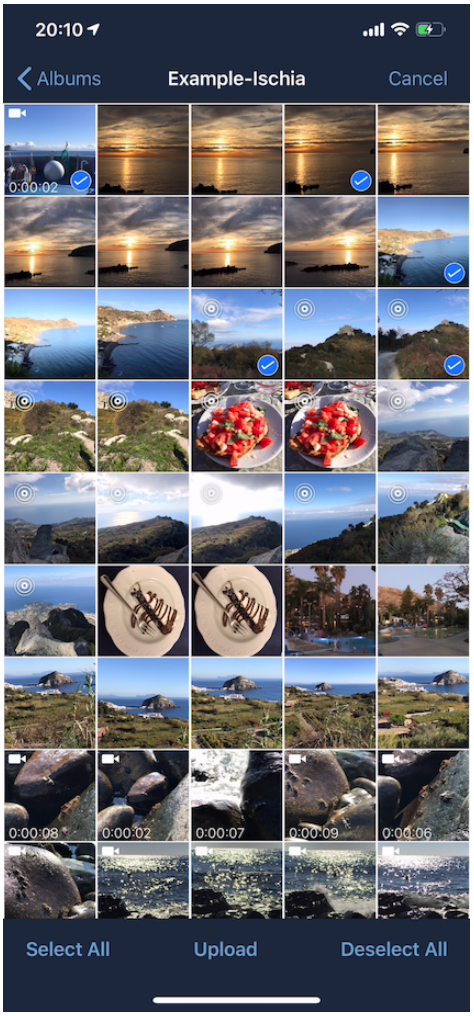
To upload files or any time from your device to your ownCloud server, click **[Upload file]**. You will then be able to select or browse through files from any app that exposes data to the iOS files app.

Make Available Offline

Please see the Offline Storage section.

Upload File From Your Photo Library

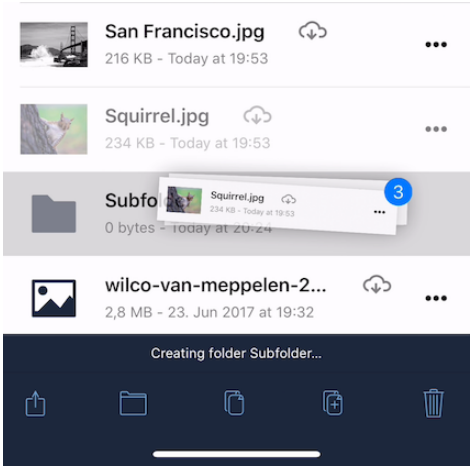
To upload photos from your photo library, you first need to allow the iOS app access to your photos. After that, you can browse through your photos, as you normally would. You can then select one or more photos by pressing them, or click **[Select All]** in the bottom left-hand corner to select all photos in the current folder. When you're happy with your photo selection, click **[Upload]** and the photo(s) will be uploaded.



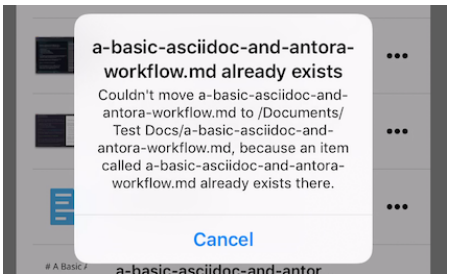
Move Files and Folders

Whether you are using the iPhone or iPad version of the ownCloud app, you can select and drag and drop one or more files and folders from one folder to another. To do so, you first press **[Select]** in the top right-hand corner and select one or more files and/or folders. Then, you press and hold on any of the selected files and folders and:

- Drag and drop them over a folder in the current directory
- Drag and drop them over the "**Move to**" icon (or tap the icon), near the bottom left-hand side of the screen. You then navigate to the folder that you want to move them to and click **[Move here]** at the bottom of the screen.



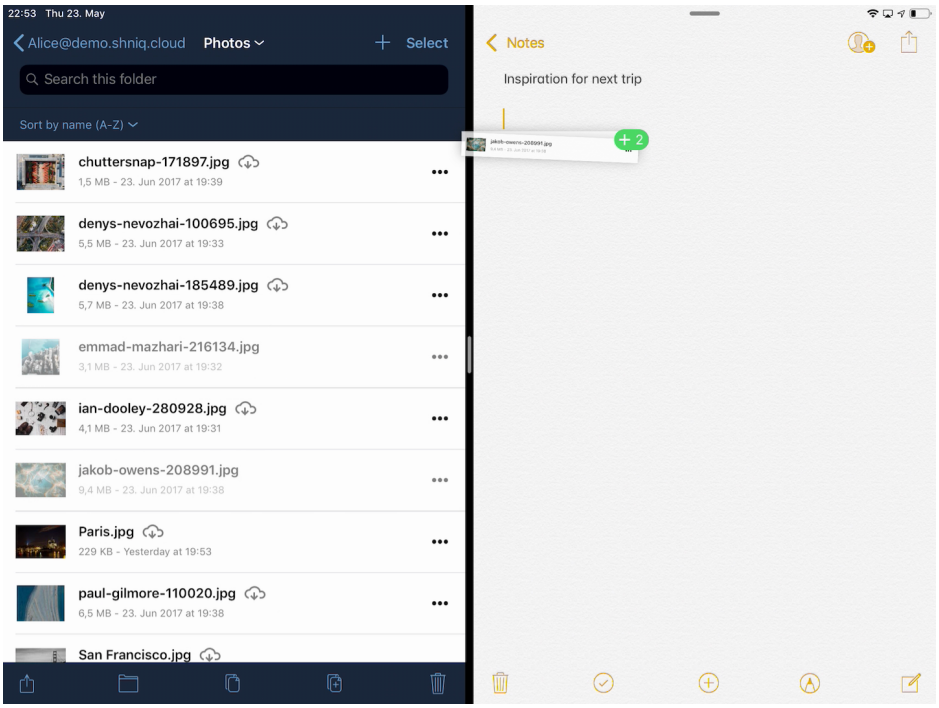
If a file or folder with the same name as one or more of those being moved, already exists in the destination directory, you will see a warning that the file or folder could not be moved.



Drag & Drop Files Between Apps (iPad-only)

The iOS app supports the multitasking features on iPad. If you open it as a second app with Slide Over, you can use two apps at the same time with Split View and drag and drop one or more files between the two apps. Refer to Apple's Multitasking On Your iPad guide for more information.

Drag and drop multiple files from ownCloud iOS App to macOS Notes



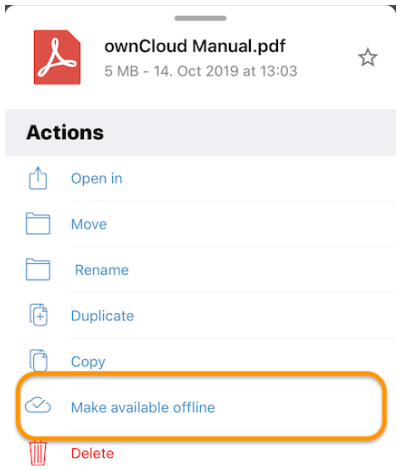
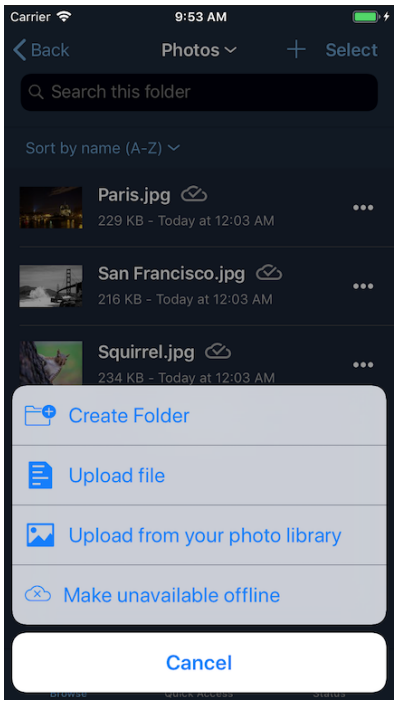
Offline Files and Folders

Introduction

From version 1.1.0, the ownCloud iOS app supports offline storage of files and folders, allowing them to be accessed even when an internet connection is temporarily unavailable.

Making a File or Folder Available Offline

To make a file or folder available offline, click on the *More* icon and press "*Make available offline*" from the list of available actions. In the screenshots below, you can see an example of the available actions for files and folders.

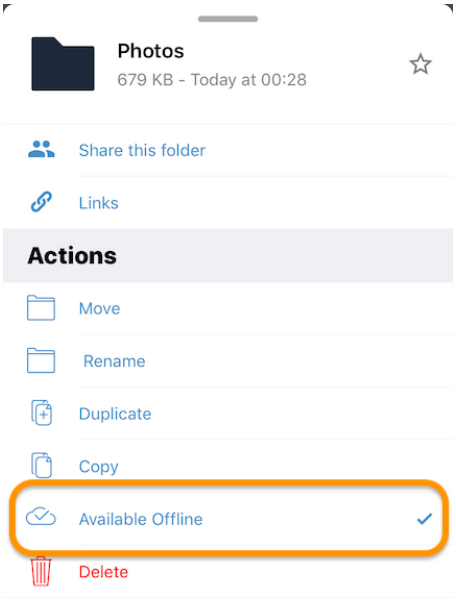
Make file available offline	Make folder available offline
	

Files that have been made available offline are identifiable by the available offline icon. You can see an example of the icon image below.
.

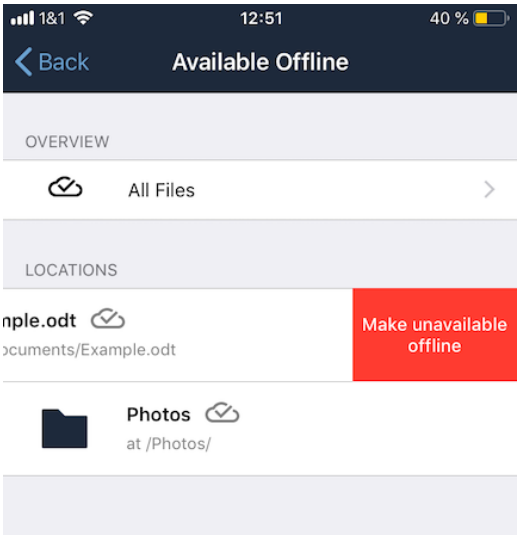
Removing a File or Folder from Offline Storage

Any file or folder that has been made available offline can also be removed from offline storage in several ways.

- By pressing the *More* icon next to a file or folder and pressing "*Available offline*" from the list of available actions.



- By swiping left on an item in the Available Offline Locations list and pressing "*Make unavailable offline*".



Viewing Offline Files

To view all offline files, from the *Quick Access* menu, tap *Available Offline*. If no files have been marked as available offline, then no files will be available.

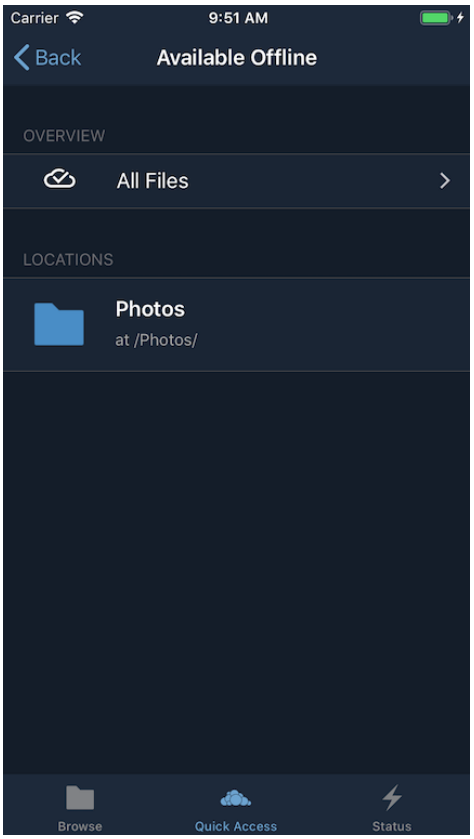
If one or more files have been marked as available offline, then you have two ways of viewing them.

- View files by location
- View a list of all files

View Offline Files by Location

In the screenshot below, you can see that there are one or more files in the *Photos* directory that have been marked as available offline. If you tap one of the available directories, you will then see all files in that directory that are available offline, similar to how you would view files normally.

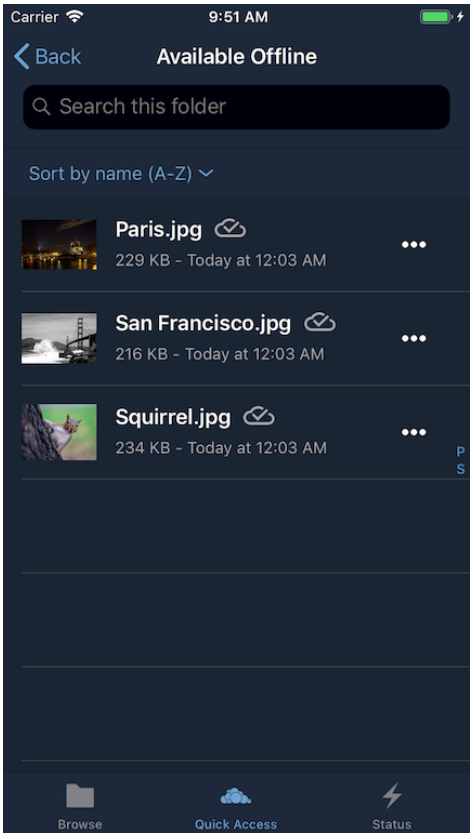
View offline files by location



View a List of All Offline Files

In the screenshot below, you can see all the items that have been marked as available offline.

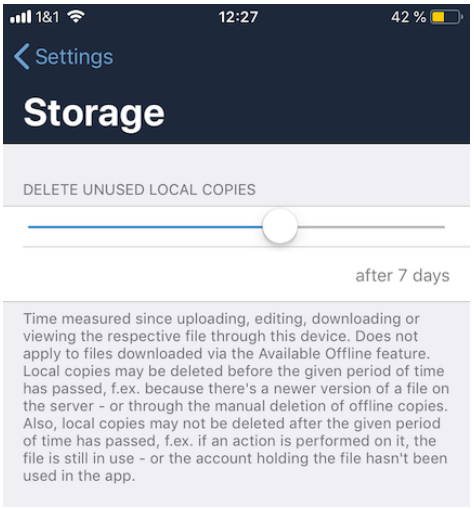
View all offline files



Storage

Locally available file copies can be set to be automatically deleted after a specified period, ranging from 1 minute to 30 days, to clean up device space. The default is seven days. This is available under **Settings › Storage › Delete unused local copies**.

This setting applies to all local files, not just available offline files.

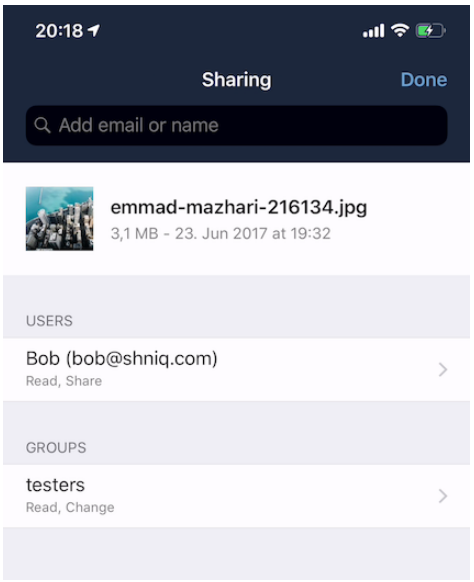


Collaboration and Links

Introduction

This section shows how to collaborate with other users and how to work with links.

Collaborate With Other Users on Files and Folders



The Mobile App for iOS supports the following collaboration functionality:

- Adding users and grant permissions
- Adding groups and grant permissions
- Changing permissions
- Deleting collaborators

Manage Links

Grant Access With Links

Just as with the web interface, the iOS app lets you grant access to both files and folders with links. Specifically, you can do the following:

- Copy private links
- Create public links
- Set passwords and expiration dates
- Send public link URLs to other apps (mail, messenger)
- Delete public links

Copy Private Links

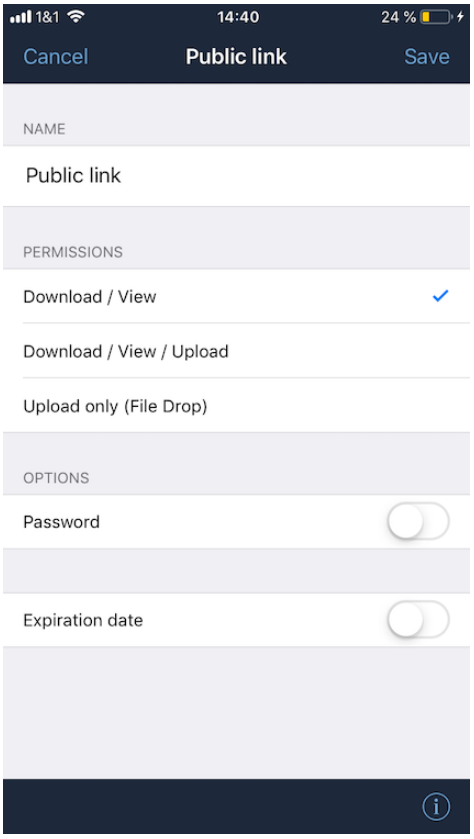
To copy a private link to a file or folder, click the more icon on the far right-hand side of the file, and then click **[Links]**. As in the example below, you will then see the private link to the file. Clicking **[copy]** on the far right-hand side will copy the link to the iOS clipboard.

Create Public Links

To create public links to a file or folder, click the more icon on the far right-hand side of the file, and then click **[Links]**. Then, under Public Links, click **[Create Public Link]** in the Public Links section. You will then see the links for the option, as in the image below.

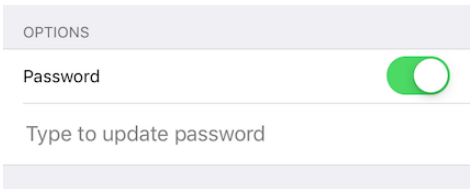
As with the web interface, you can set:

- A link name
- Link permissions
- A link password and expiration date

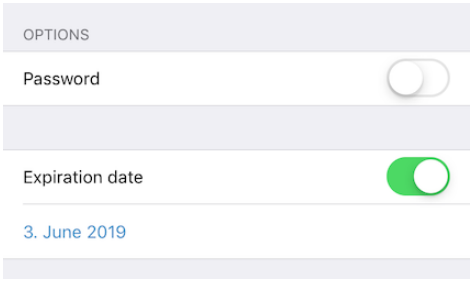


To see more details about each option, click the info icon in the bottom right-hand corner.

Set Passwords and Expiration Dates



To set a password on a public link, under "*Options*", enable **[Password]**. Then, type a password in the field that appears below the Password option.



To set an expiration date on a public link, under "*Options*", enable **[Expiration date]**. Then, pick the date that the link should expire with the date picker that appears below the Expiration date option.

Send Public Link URLs to Other Apps

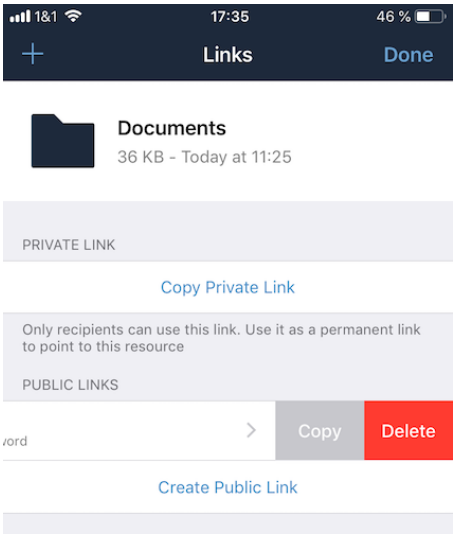
To share a public link URL via other apps:

1. Open the Public Link's details.
2. Click the share button at the bottom left-hand corner, which opens the iOS Share Sheet.
3. Share the link through the app of your choice.

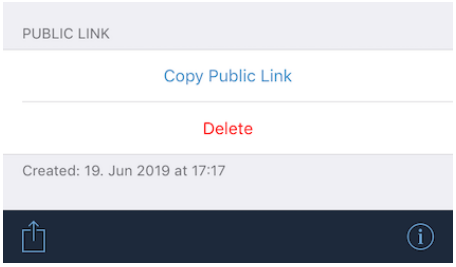
Delete Public Links

There are two ways to delete a public link.

1. When viewing the list of links for a file or folder, swipe left on the link that you want to delete, and click **[Delete]**.



2. When viewing the Public Link, click **[Delete]** at the bottom of the page, under **[Copy Public Link]**.



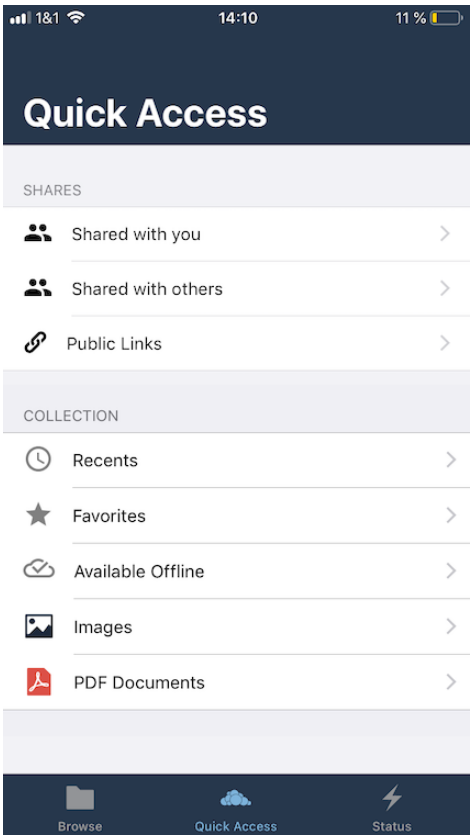
View Public Links

The quickest way to view Public Links is by navigating to **Quick Access › Shares › Public Links**. There, you will see a list of files and folders that have Public Links. For any of the files and folders, click the More icon, where you will see the number of links for that file or folder.

Quick Access

Introduction

The app allows you to quickly access the key aspects of the application, by clicking **[Quick Access]**, located in the footer of the application.



Quick access is broken down into two sections: *Shares* and *Collection*.

Shares

Shares, as the name implies, lets you quickly see all files that are:

Shared with you

See a list of all resources you accepted from other users.

Shared with others

See a list of all resources where you invited collaborators.

Public links

See a list of all resources where you created links.

Collection

Collection gives you quick access to:

Recently accessed files

These are files that were used sometime within the previous week, where used covers when it was imported, locally update, or was download.

Files marked as favorite

See a list of all resources in your account set as favourite.

Files available offline

See a list of all files and folders in your account that are available offline.

Image files

See a list of all image files in your account.

PDF files

See a list of all PDF files in your account.

iOS App and iOS Files App Integration and 3rd Party Apps

Introduction

The Mobile App for iOS integrates with the iOS Files App. When enabled, you can open, edit, save, and delete files and folders. However, the functionality needs to first be enabled.

Work With Your ownCloud Data in iOS Files App

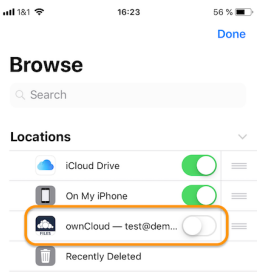
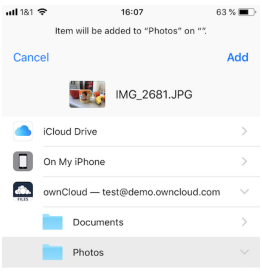
To enable this possibility, click **[Browse]**, then **[Edit]**. Next, in the list of apps under Locations, enable the ownCloud app for your account.

In future versions, it is expected that the following functionality will also be supported:

- Share (UI)
- Offline (UI)
- Favourites
- Tags

Share From Other Apps (via iOS Files App)

When the iOS app is installed and at least one account is properly configured, files on your iOS device can be shared with your ownCloud server. To do so, first enable the location from the Locations list (step through how to do this). After that, when sharing a file, first click **[Save to Files]**, choose a location to store the file, and click **[Add]**. These steps are shown in the three images below.

Enable the ownCloud app as an iOS Files app location	Pick a file to "Save to Files"	Pick the location to save the file and click "Add"
		

Task Scheduling

Introduction

Background tasks are scheduled based on the app's current context, and that context is based on a combination of factors. These factors include:

- Is the app backgrounded or in the foreground?
- Is WiFi available?
- Is the device in Low Power Mode?
- Is the device connected to external power?
- Has the photo library changed?

Tasks

Currently, two tasks are available:

- **Instant photo uploads:** This task is triggered when:
  - The photo library has new photos, and;
  - The user is connected to a WiFi network
- **Bookmark update task:** This task is triggered by the background fetch event scheduled by iOS

Instant Photo Upload

There are some things to be aware of, regarding Instant Photo Upload.

- Photo and video upload can be enabled and disabled separately.
- Before instant media upload can begin, the account and upload path have to be selected
- Changes in the photo library are detected when the app goes into the foreground.
- If a user removes the folder specified for instant upload, the upload task silently exits.
- When a user first activates instant upload, the timestamp of the activation is stored and compared against the creation date of the assets to be uploaded. As soon as one asset is successfully uploaded, its creation timestamp is used to update the instant upload activation timestamp. This:
  - Prevents assets being uploaded twice
  - Removes the need to keep track of uploaded items in a database
  - Please note, that already uploaded and edited asset is not going to be uploaded again
- Media conversion settings are taken into account by instant upload feature.

Security

Introduction

This document provides an overview of the security considerations and features in the new ownCloud iOS SDK and new ownCloud iOS App.

Authentication

API

The ownCloud iOS SDK (2018) is a general-purpose API for implementing passphrase- and token-based authentication methods, which provides three key benefits. These are:

1. Ensures structural separation of code between general connection-handling and authentication.
2. Simplifies code reviews, by limiting each implementation to one class each (e.g., OAuth2, BasicAuth).
3. Ensures extensibility.

Secrets

Authentication secrets contain information such as *usernames*, *passwords* or *tokens*, and are generated by the respective authentication method implementation. Authentication Secrets are securely stored in the app's Keychain and tagged as `AccessibleAfterFirstUnlock`. They cannot be accessed after a restart until the device has been unlocked once by the user.

Supported Methods

Authentication method implementations are available for OAuth2 and Basic Authentication.

Method Selection

After performing auto-detection, identified authentication methods are filtered and ranked by preference. The method with the highest ranking is then picked for the user.

By default, all detected methods are considered and OAuth2 ranks higher than Basic Authentication.

Filtering and ranking can be customized by MDM Configuration. This, for example, allows making OAuth2 the only possible authentication method, so no credentials need to be stored on the device.

OAuth2 Implementation

The OAuth2 implementation uses `SFAuthenticationSession`, which is described as a best practice by `RFC 8252` - when running under iOS 11. Under iOS 12, the OAuth2 implementation uses `ASWebAuthenticationSession`, which is the successor of `SFAuthenticationSession`. Benefits of using

these APIs include:

- **Privilege separation:** web content is run in a separate process
- **Trustworthiness:** apps can't inject code into or access the contents of the web view
- **Convenience for the user:** cookies from Safari are available to the web content inside the session

Connections

URL Limits

Using MDM Configuration, server URLs can be pre-filled or "hard-coded" as the only allowed server URL.

Redirects

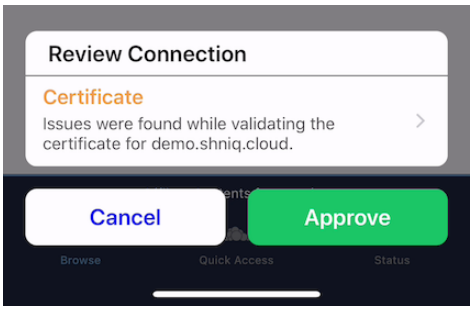
Redirects during login are not followed silently. Instead, they are reported to the user and must be explicitly approved.

SSL/TLS Certificates

When adding servers, users have the opportunity to view a detailed summary of the server's SSL/TLS certificate before they are prompted for credentials or authentication via OAuth2.

If a SSL/TLS certificate fails trust evaluation (e.g., because it's self-signed or signed by an unknown Certificate Authority), the user is given an opportunity to:

- View a detailed summary of the certificate, by clicking on the notification.
- Trust the certificate, despite the warnings, by clicking **[Approve]**.
- Reject the certificate, by clicking **[Cancel]**.



In the case of redirects across several HTTPS servers, users are given the opportunity to review the certificates of all servers involved in addition to the redirects.

MDM Configuration support is planned for:

- Pre-approving specific certificates and certificates with specific public keys.
- Allowing only connections with specific certificates or certificates with specific public keys.

Inspecting Certificate Details

If users want to inspect the details of an approved security certificate, from the Accounts list, swipe left on the account that you want to check the certificate of and click **[Edit]**. Then, click the row **[Certificate Details]**. You will then see the certificate's details, starting with the validation status.

More Information

For more information, please refer to the security information.

Data

Separation

Separate directories are used for the data of every server connection. This provides the following benefits:

- A strong barrier against accidentally spilling data between different connections.
- All data relating to a connection can be deleted by deleting the respective directory.

Encryption

The app uses the filesystem encryption built into iOS. Using the `CompleteUntilFirstUserAuthentication` file protection, data can't be accessed after a restart until the device has been unlocked once by the user.

Sync

The Sync Strategies, planned to be used in the app, focus on preventing data loss locally and remotely.

Secure Document View

HTML and Microsoft Office document content is viewed using `WKWebView`, which renders the content in a separate process. Additional hardening is achieved by disabling JavaScript and blocking all network requests, which protects against lesser known, non-obvious attacks like CSS Keylogging.

Passcode

Users can set a Passcode to control access to the app. Find out more about this in the Passcode section of the Settings documentation.

Miscellaneous

Continuous Integration (CI)

Continuous Integration tests verify that central security mechanisms and assumptions work as

expected, covering areas such as *redirections*, *certificate handling*, common Man-in-the-middle (MITM) attack scenarios, and the secure storage of authentication secrets.

SQL Injection

To protect against SQL injection attacks, parameters are never made part of the SQL statements themselves. Instead, placeholders are used and the parameters are subsequently bound to the SQL statements. For example, instead of running a query, such as `SELECT * FROM users WHERE name='John Doe'`, the query would be parameterised, such as: `SELECT * FROM users WHERE name=:nameToSearchFor`.

Reproducibility

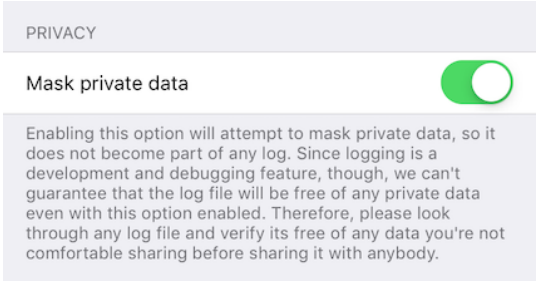
The build script that created the OpenSSL binaries used in the app is available in the SDK's GitHub repository and can be used to reproduce the build result.

OpenSSL is used solely to provide detailed summaries of SSL/TLS certificates - functionality that iOS is currently missing.

Planned Logging Feature (not included in released yet!!)

When logging information, parts of the log message can be tagged as private. If "**Mask private data**" is enabled, under **Settings › Logging** (it is by default), these parts will be - before the log message is written - either replaced with `«private»` or a trimmed version that doesn't contain privacy-sensitive information.

An example for the latter would be an `NSError` object's error message containing the names of the item it is about. If masked, only the error's error domain and error code are written to the log, but not the error message.



iOS Frequently Asked Questions (FAQ)

Introduction

Here you can find some of the most frequently asked questions about the ownCloud iOS app.

Usage

Compatibility With ownCloud Server Older Than Version 10.0

Previous ownCloud iOS app with support for ownCloud Server < 10.0 is still available in App Store.

Compatibility With iOS Versions Older Than iOS 12.0

Previous ownCloud iOS app with support for iOS < 12.0 is still available in App Store.

Login With TOTP and Other 2FA

ownCloud server must have the OAuth2 app installed, configured, and enabled to use Two-Factor Authentication. Please contact your ownCloud administrator for more details.

Feature Requests

FileProvider: Access Full Folder, Not Only Single Files

This seems like a bigger challenge for us. We track the status here:
https://github.com/owncloud/ios-app/issues/604

FileProvider: Support "Recent items" and "Favorites"

To support "Recent items" and "Favorites" in the iOS Files app, implementation of the Working Set is needed. This is one of the next roadmap items, but there's no ETA yet.

Text Editing Inside the iOS App

Currently this isn't a planned feature. Focus is more to build the best integration in serious 3rd party text editing app.

Here you can find previous discussions:
https://github.com/owncloud/ios-app/issues/317

Missing Translations or Translation Bugs

You can help translate in your language or fix a bug.

We use Transifex for translations. Please register here for an account and join the global community.

Testing New Features

Use this link to join the beta program on iOS.

Appendices

In this section, you find supporting information.

Mobile Device Management (MDM)

Introduction

Starting with iOS 7, Apple added support for managed application configuration. An MDM server can push a configuration to the iOS App. The app can access this configuration (read-only) using the `NSUserDefaults` class by reading a configuration dictionary under the key *com.apple.configuration.managed*. An app can also observe a system notification (*NSUserDefaultsDidChangeNotification*) to get notified about configuration changes. In addition feedback can be queried back by MDM server. To enable that, app has to write a dictionary with feedback information into user defaults under *com.apple.feedback.managed* key. The configuration is basically a key-value dictionary provided as a `.plist` file.

Configurable Settings

ownCloud App implements a mechanism internally called Class Settings which can be derived from different sources:

- Environment variables which e.g. can be set in Xcode for testing. In this case setting keys have to be prepended with *oc:* prefix.
- User preferences accessed by the very same API but stored under *org.owncloud.user-settings* key.
- Settings dictionary pushed by an MDM Server and accessible using `NSUserDefaults` API under the key *com.apple.configuration.managed*.
- Default settings defined directly in the app sourcecode.
- Branding.plist which is the part of the Xcode project under ownCloud/Resources/Theming. It allows to override class settings by specifying them in the `Configuration` section

This is also an order in which these settings take precedence (environment variables have highest priority). So, when settings are accessed, they are merged and higher priority value for the same key overwrites lower priority ones.

Some settings are accessed only once at runtime and the read value is cached, so that new setting to take effect may a require an app to be terminated and restarted.

App Basic Configurations

There are few settings allowing to mark an app installation as BETA and e.g. to supress UIKit animation and review prompt.

Key	Type	Default	Description	Status
app.app-store-link	string	https://itunes.apple.com/app/id1359583808?mt=8	URL for the app in the App Store.	advanced candidate
app.enable-review-prompt	bool	true	Enable/disable review prompt.	advanced candidate
app.recommend-to-friend-enabled	bool	true	Enables/disables the recommend to a friend entry in the settings.	advanced candidate
app.enable-ui-animations	bool	true	Enable/disable UI animations.	debugOnly
app.is-beta-build	bool	false	Controls if the app is built for beta or release purposes.	debugOnly
app.show-beta-warning	bool	false	Controls whether a warning should be shown on the first run of a beta version.	debugOnly

Extensions / Actions

ownCloud app uses internally a plug-in like mechanism called extensions. Extensions are used to implement menu actions mostly found under "+" menu allowing to add new items (Upload media, take photo etc.) or in more menu (Copy, Move, Open in etc.). Using below settings actions / extensions can be disabled. Extensions are enabled by default, however this might depend on licensing requirements of a particular extension.

<table>
<tr><th>Key</th><th>Type</th><th>Default</th><th>Description</th><th>Status</th></tr>
<tr><td>action.allowed</td><td>stringArray</td><td>[]</td><td>List of all allowed actions. If provided, actions not listed here are not allowed.

<table><tr><td>Value</td><td>Description</td></tr><tr><td>com.owncloud.action.collaborate</td><td>Sharing</td></tr><tr><td>com.owncloud.action.copy</td><td>Copy</td></tr><tr><td>com.owncloud.action.createFolder</td><td>Create folder</td></tr><tr><td>com.owncloud.action.cutpasteboard</td><td>Cut</td></tr><tr><td>com.owncloud.action.delete</td><td>Delete</td></tr><tr><td>com.owncloud.action.duplicate</td><td>Duplicate</td></tr><tr><td>com.owncloud.action.favorite</td><td>Favorite item</td></tr><tr><td>com.owncloud.action.importpasteboard</td><td>Paste</td></tr><tr><td>com.owncloud.action.links</td><td>Links</td></tr></table></td><td>advanced candidate</td></tr>
</table>

<table>
<tr><th>Key</th><th>Type</th><th>Default</th><th>Description</th><th>Status</th></tr>
<tr><td>action.disallowed</td><td>stringArray</td><td>[]</td><td>List of all disallowed actions. If provided, actions not listed here are allowed.

<table><tr><th>Value</th><th>Description</th></tr><tr><td>com.owncloud.action.collaborate</td><td>Sharing</td></tr><tr><td>com.owncloud.action.copy</td><td>Copy</td></tr><tr><td>com.owncloud.action.createFolder</td><td>Create folder</td></tr><tr><td>com.owncloud.action.cutpasteboard</td><td>Cut</td></tr><tr><td>com.owncloud.action.delete</td><td>Delete</td></tr><tr><td>com.owncloud.action.duplicate</td><td>Duplicate</td></tr><tr><td>com.owncloud.action.favorite</td><td>Favorite item</td></tr><tr><td>com.owncloud.action.importpasteboard</td><td>Paste</td></tr><tr><td>com.owncloud.action.links</td><td>Links</td></tr></table></td><td>advanced candidate</td></tr>
</table>

(*) These extensions might require additional license (in-app purchase, enterprise version).

Display Settings

To customize file list UI behevior, following settings are available:

Passcode Enforcement

If your organization policies require users to use a passcode as an additional security barrier for managed apps, the below setting will allow to enforce this requirement.

Key	Type	Default	Description	Status
passcode.enforced	bool	`false`	Controls wether the user MUST establish a passcode upon app installation	advanced `candidate`
passcode.maximumPasscodeDigits	int	`6`	Controls how many passcode digits are maximal possible for passcode lock.	advanced `candidate`
passcode.requiredPasscodeDigits	int	`4`	Controls how many passcode digits are at least required for passcode lock.	advanced `candidate`

Bookmark

Below settings allow to configure the app to use a certain server URL and even bind it to this URL only by setting the default non-editable.

Key	Type	Default	Description	Status
bookmark.default-url	string		The default URL for the creation of new bookmarks.	supported `candidate`

Key	Type	Default	Description	Status
bookmark.url-editable	bool	`true`	Controls whether the server URL in the text field during the creation of new bookmarks can be changed.	supported `candidate`

Item Policies

Key	Type	Default	Description	Status
item-policy.local-copy-expiration	int	`604800`	The number of seconds that a file hasn't been downloaded, modified or opened after which the local copy is removed.	advanced `candidate`
item-policy.local-copy-expiration-enabled	bool	`true`	Controls whether local copies should automatically be removed after they haven't been downloaded, modified or opened for a period of time.	advanced `candidate`
item-policy.vacuum-sync-anchor-ttl	bool	`60`	Number of seconds since the removal of an item after which the metadata entry may be finally removed.	debugOnly

Connection

Settings concerinng HTTP user agent, cookies, background support etc.

Key	Type	Default	Description	Status
connection.allow-cellular	bool	`true`	Allow the use of cellular connections.	recommended `candidate`

Key	Type	Default	Description	Status
core.cookie-support-enabled	bool	`true`	Enable or disable per-process, in-memory cookie storage.	supported `candidate`
http.user-agent	string	`ownCloudApp/{{app.version}} ({{app.part}}/{{app.build}}; {{os.name}}/{{os.version}}; {{device.model}})`	A custom `User-Agent` to send with every HTTP request. The following placeholders can be used to make it dynamic: - `{{app.build}}`: the build number of the app (f.ex. `123`) - `{{app.version}}`: the version of the app (f.ex. `1.2`) - `{{app.part}}`: the part of the app (more exactly: the name of the main bundle) from which the request was sent (f.ex. `App`, `ownCloud File Provider`) - `{{device.model}}`: the model of the device running the app (f.ex. `iPhone`, `iPad`) - `{{device.model-id}}`: the model identifier of the device running the app (f.ex. `iPhone8,1`) - `{{os.name}}` : the name of the operating system running on the device (f.ex. `iOS`, `iPadOS`) - `{{os.version}}`: the version of operating system running on the device (f.ex. `13.2.2`)	supported `candidate`

<table>
<tr><th>Key</th><th>Type</th><th>Default</th><th>Description</th><th>Status</th></tr>
<tr><td>connection.always-request-private-link</td><td>bool</td><td><code>false</code></td><td>Controls whether private links are requested with regular PROPFINDs.</td><td>advanced <code>candidate</code></td></tr>
<tr><td>connection.plain-http-policy</td><td>string</td><td><code>warn</code></td><td>Policy regarding the use of plain (unencryped) HTTP URLs for creating bookmarks. A value of <code>warn</code> will create an issue (typically then presented to the user as a warning), but ultimately allow the creation of the bookmark. A value of <code>forbidden</code> will block the use of <code>http</code>-URLs for the creation of new bookmarks.</td><td>advanced <code>candidate</code></td></tr>
<tr><td>connection.validator-flags</td><td>stringArray</td><td></td><td>Allows fine-tuning the behavior of the connection validator by enabling/disabling aspects of it.<table><tr><th>Value</th><th>Description</th></tr><tr><td><code>502-triggers</code></td><td>Connection validation is triggered when receiving a responses with 502 status.</td></tr><tr><td><code>clear-cookies</code></td><td>Clear all cookies for the connection when entering connection validation.</td></tr></table></td><td>advanced <code>candidate</code></td></tr>
</table>

Key	Type	Default	Description	Status
core.action-concurrency-budgets	dictionary	`map[actions:10 all:0 download:3 download-wifi-and-cellular:3 download-wifi-only:2 transfer:6 upload:3 upload-cellular-and-wifi:3 upload-wifi-only:2]`	Concurrency budgets available for sync actions by action category.	advanced `candidate`
core.scan-for-changes-interval	int	`10`	Minimum number of seconds until the next scan for changes, measured from the completion of the previous scan.	advanced `candidate`
connection.allow-background-url-sessions	bool	`true`	Allow the use of background URL sessions. Note: depending on iOS version, the app may still choose not to use them. This settings is overriden by `force-background-url-sessions`.	debugOnly
connection.force-background-url-sessions	bool	`false`	Forces the use of background URL sessions. Overrides `allow-background-url-sessions`.	debugOnly
connection.minimum-server-version	string	`10.0`	The minimum server version required.	debugOnly

Key	Type	Default	Description	Status
core.override-availability-signal	bool		Override the availability signal, so the host is considered to always be in maintenance mode (`true`) or never in maintenance mode (`false`).	debugOnly
core.override-reachability-signal	bool		Override the reachability signal, so the host is always considered reachable (`true`) or unreachable (`false`).	debugOnly
core.thumbnail-available-for-mime-type-prefixes	stringArray	`[*]`	Provide hints that thumbnails are available for items whose MIME-Type starts with any of the strings provided in this array. Providing an empty array turns off thumbnail loading. Providing `["*"]` turns on thumbnail loading for all items.	debugOnly

<table>
<tr><th>Key</th><th>Type</th><th>Default</th><th>Description</th><th>Status</th></tr>
<tr><td>host-simulator.active-simulations</td><td>stringArray</td><td>[]</td><td>Active Host simulation extensions.<table><tr><td>Value</td><td>Description</td></tr><tr><td>five-seconds-of-404</td><td>Return status code 404 for every request for the first five seconds.</td></tr><tr><td>only-404</td><td>Return status code 404 for every request.</td></tr><tr><td>recovering-apm</td><td>Redirect any request without cookies to a bogus endpoint for 30 seconds, then to a cookie-setting endpoint, where cookies are set - and then redirect back.</td></tr><tr><td>reject-downloads-500</td><td>Reject Downloads with status 500 responses.</td></tr><tr><td>simple-apm</td><td>Redirect any request without cookies to a cookie-setting endpoint, where cookies are set - and then redirect back.</td></tr></table></td><td>debugOnly</td></tr>
</table>

Server Endpoints

Individually configurable endpoints of the ownCloud server instance.

Key	Type	Default	Description	Status
connection.endpoint-capabilities	string	`ocs/v2.php/cloud/capabilities`	Endpoint to use for retrieving server capabilities.	advanced `candidate`
connection.endpoint-recipients	string	`ocs/v2.php/apps/files_sharing/api/v1/sharees`	Path of the sharing recipient API endpoint.	advanced `candidate`
connection.endpoint-remote-shares	string	`ocs/v2.php/apps/files_sharing/api/v1/remote_shares`	Path of the remote shares API endpoint.	advanced `candidate`
connection.endpoint-shares	string	`ocs/v2.php/apps/files_sharing/api/v1/shares`	Path of the shares API endpoint.	advanced `candidate`
connection.endpoint-status	string	`status.php`	Endpoint to retrieve basic status information and detect an ownCloud installation.	advanced `candidate`
connection.endpoint-thumbnail	string	`index.php/apps/files/api/v1/thumbnail`	Path of the thumbnail endpoint.	advanced `candidate`

Key	Type	Default	Description	Status
connection.endpoint-user	string	`ocs/v2.php/cloud/user`	Endpoint to use for retrieving information on logged in user.	advanced `candidate`
connection.endpoint-webdav	string	`remote.php/dav/files`	Endpoint to use for WebDAV.	advanced `candidate`
connection.endpoint-webdav-meta	string	`remote.php/dav/meta`	Endpoint to use for WebDAV metadata.	advanced `candidate`
connection.well-known	string	`.well-known`	Path of the .well-known endpoint.	advanced `candidate`

Connection Authentication / Security

Settings concerning certificate validation policies.

<table>
<tr><th>Key</th><th>Type</th><th>Default</th><th>Description</th><th>Status</th></tr>
<tr><td>connection.allowed-authentication-methods</td><td>stringArray</td><td></td><td>Array of allowed authentication methods. Nil/Missing for no restrictions.<table><tr><th>Value</th><th>Description</th></tr><tr><td><code>com.owncloud.basicauth</code></td><td>Basic Auth</td></tr><tr><td><code>com.owncloud.oauth2</code></td><td>OAuth2</td></tr><tr><td><code>com.owncloud.openid-connect</code></td><td>OpenID Connect</td></tr></table></td><td>recommended <code>candidate</code></td></tr>
<tr><td>connection.preferred-authentication-methods</td><td>stringArray</td><td><code>[com.owncloud.openid-connect com.owncloud.oauth2 com.owncloud.basicauth]</code></td><td>Array of authentication methods in order of preference (most preferred first).<table><tr><th>Value</th><th>Description</th></tr><tr><td><code>com.owncloud.basicauth</code></td><td>Basic Auth</td></tr><tr><td><code>com.owncloud.oauth2</code></td><td>OAuth2</td></tr><tr><td><code>com.owncloud.openid-connect</code></td><td>OpenID Connect</td></tr></table></td><td>recommended <code>candidate</code></td></tr>
</table>

Key	Type	Default	Description	Status
connection.certificate-extended-validation-rule	string	`bookmarkCertificate == serverCertificate`	Rule that defines the criteria a certificate needs to meet for OCConnection to recognize it as valid for a bookmark. Examples of expressions: - `bookmarkCertificate == serverCertificate`: the whole certificate needs to be identical to the one stored in the bookmark during setup. - `bookmarkCertificate.publicKeyData == serverCertificate.publicKeyData`: the public key of the received certificate needs to be identical to the public key stored in the bookmark during setup. - `serverCertificate.passedValidationOrIsUserAccepted == true`: any certificate is accepted as long as it has passed validation by the OS or was accepted by the user. - `serverCertificate.commonName == "demo.owncloud.org"`: the common name of the certificate must be "demo.owncloud.org". - `serverCertificate.rootCertificate.commonName == "DST Root CA X3"`: the common name of the root certificate must be "DST Root CA X3". -	advanced `candidate`

Key	Type	Default	Description	Status
connection.renewed-certificate-acceptance-rule	string	`(bookmarkCertificate.publicKeyData == serverCertificate.publicKeyData) OR check.parentCertificatesHaveIdenticalPublicKeys == true) AND (serverCertificate.passedValidationOrIsUserAccepted == true OR bookmarkCertificate.parentCertificate.sha256Fingerprint.asFingerPrintString == "73 0C 1B DC D8 5F 57 CE 5D C0 BB A7 33 E5 F1 BA 5A 92 5B 2A 77 1D 64 0A 26 F7 A4 54 22 4D AD 3B") AND (bookmarkCertificate.rootCertificate.sha256Fingerprint.asFingerPrintString == "06 87 26 03 31 A7 24 03 D9 09 F1 05 E6 9B CF 0D 32 E1 BD 24 93 FF C6 D9 20 6D 11 BC D6 77 07 39") AND (serverCertificate.parentCertificate.sha256Fingerprint.asFingerPrintString == "67 AD D1 16 6B 02 0A E6 1B 8F 5F C9 68 13 C0 4C 2A A5 89 96 07 96 86 55 72 A3 C7 E7 37 61 3D FD") AND (serverCertificate.rootCertificate.sha256Fingerprint.asFingerPrintString == "96 BC EC 06 26 49 76 F3 74 60 77 9A CF 28 C5 A7 CF E8 A3 C0 AA E1 1A 8F FC EE 05 C0 BD DF 08 C6") AND (serverCertificate.passedValidationOrIsUserAccepted == true`	Rule that defines the criteria that need to be met for OCConnection to accept a renewed certificate and update the bookmark's certificate automatically instead of prompting the user. Used when the extended validation rule fails. Set this to `never` if the user should always be prompted when a server's certificate changed.	advanced `candidate`

Key	Type	Default	Description	Status
user-settings.allow	stringArray		List of settings (as flat identifiers) users are allowed to change. If this list is specified, only these settings can be changed by the user.	advanced `candidate`
user-settings.disallow	stringArray		List of settings (as flat identifiers) users are not allowed to change. If this list is specified, all settings not on the list can be changed by the user.	advanced `candidate`
connection.transparent-temporary-redirect	bool	`false`	Controls whether 307 redirects are handled transparently at the HTTP pipeline level (by resending the headers and body).	debugOnly

OAuth2 Based Authentication

Settings allowing to configure OAuth2 based authentication.

Key	Type	Default	Description	Status
authentication-oauth2.oa2-authorization-endpoint	string	`index.php/apps/oauth2/authorize`	OAuth2 authorization endpoint.	advanced `candidate`

Key	Type	Default	Description	Status
authentication-oauth2.oa2-client-id	string	`mxd5OQDk6es5LzOzRvidJNfXLUZS2oN3oUFeXPP8LpPrhx3UroJFduGEYIBOxkY1`	OAuth2 Client ID.	advanced `candidate`
authentication-oauth2.oa2-client-secret	string	`KFeFWWEZO9TkisIQzR3fo7hfiMXlOpaqP8CFuTbSHzV1TUuGECglPxpiVKJfOXIx`	OAuth2 Client Secret.	advanced `candidate`
authentication-oauth2.oa2-redirect-uri	string	`oc://ios.owncloud.com`	OAuth2 Redirect URI.	advanced `candidate`
authentication-oauth2.oa2-token-endpoint	string	`index.php/apps/oauth2/api/v1/token`	OAuth2 token endpoint.	advanced `candidate`

Key	Type	Default	Description	Status
authentication-oauth2.oa2-expiration-override-seconds	int		OAuth2 Expiration Override - lets OAuth2 tokens expire after the provided number of seconds (useful to prompt quick `refresh_token` requests for testing)	debugOnly

Logging

Logging settings control the ammount and type of app internal log messages stored as text files and accessible via settings menu.

Key	Type	Default	Description	Status
log.level	int	`4`	Log level Value: Description `-1`: verbose `0`: debug `1`: info `2`: warning `3`: error `4`: off	supported `candidate`
log.privacy-mask	bool	`false`	Controls whether certain objects in log statements should be masked for privacy.	supported `candidate`
log.blank-filtered-messages	bool	`false`	Controls whether filtered out messages should still be logged, but with the message replaced with `-`.	advanced `candidate`
log.colored	bool	`false`	Controls whether log levels should be replaced with colored emojis.	advanced `candidate`

<table>
<tr><th>Key</th><th>Type</th><th>Default</th><th>Description</th><th>Status</th></tr>
<tr><td>log.enabled-components</td><td>stringArray</td><td>[writer.stderr writer.file]</td><td>List of enabled logging system components.<table><tr><td>Value</td><td>Description</td></tr><tr><td>option.log-file-operations</td><td>Log internal file operations</td></tr><tr><td>option.log-requests-and-responses</td><td>Log HTTP requests and responses</td></tr><tr><td>writer.file</td><td>Log file</td></tr><tr><td>writer.stderr</td><td>Standard error output</td></tr></table></td><td>advanced candidate</td></tr>
<tr><td>log.format</td><td>string</td><td>text</td><td>Determines the format that log messages are saved in<table><tr><td>Value</td><td>Description</td></tr><tr><td>json</td><td>Detailed JSON (one line per message).</td></tr><tr><td>json-composed</td><td>A simpler JSON version where details are already merged into the message.</td></tr><tr><td>text</td><td>Standard logging as text.</td></tr></table></td><td>advanced candidate</td></tr>
<tr><td>log.maximum-message-size</td><td>int</td><td>0</td><td>Maximum length of a log message before the message is truncated. A value of 0 means no limit.</td><td>advanced candidate</td></tr>
</table>

Key	Type	Default	Description	Status
log.omit-matching	stringArray		If set, omits logs messages containing any of the exact terms in this array.	advanced `candidate`
log.omit-tags	stringArray		If set, omits all log messages tagged with tags in this array.	advanced `candidate`
log.only-matching	stringArray		If set, only logs messages containing at least one of the exact terms in this array.	advanced `candidate`
log.only-tags	stringArray		If set, omits all log messages not tagged with tags in this array.	advanced `candidate`
log.single-lined	bool	`true`	Controls whether messages spanning more than one line should be broken into their individual lines and each be logged with the complete lead-in/lead-out sequence.	advanced `candidate`
log.synchronous	bool	`false`	Controls whether log messages should be written synchronously (which can impact performance) or asynchronously (which can loose messages in case of a crash).	advanced `candidate`
measurements.enabled	bool	`true`	Turn measurements on or off	debugOnly

AppConfig XML Schema

The XML format, developed by AppConfig community, makes it easy for developers to define and deploy an app configuration. It not only supports configuration variables having default values, but also provides a configuration UI description, which can be interpreted by the tool and which generates a plist file. Moreover, specfile XML is consistently supported by major EMM vendors.

AppConfig conformant spec file tailored to administrator needs and containing one or more of the

above settings can be easily created using Config Spec Creator tool hosted at AppConfig website.

Example: Deployment With MobileIron

1. Open AppConfig Generator
2. Upload a specfile.xml.
3. Change the configuration options.
4. Download the generated plist file (ManagedAppConfig).
5. Open MobileIron Core.
6. Navigate to **Policies and Configs › Add New › Apple › iOS/tvOS › Managed App Config**
7. Upload the generated plist and specify name, bundle ID, and description

Example: Deployment With Jamf Pro

1. Open AppConfig Generator
2. Upload a specfile.xml.
3. Change the configuration options.
4. Copy Dictionary (button).
5. Open Jamf Pro.
6. Navigate to **Devices › Mobile Device Apps › ownCloud - File Sync and Share › iOS/tvOS › App Configuration › Edit**
7. Paste the generated Dictionary into the "Preferences" field.

References

- Introducing The AppConfig Community
- Mobile Device Management Protocol Reference

Troubleshooting

Introduction

If you experience problems while using the iOS app, you can use this guide to hopefully find a solution.

Logging

Locating App Logs and iOS App Crash Logs

There are two kinds of logs recorded in different locations:

The Log Created by the iOS App

See Capturing App Debug Logs for how to enable app logs. On the same screen where you enable logging, you can access the log files. Touch **Share log file** which opens a new screen with all the log files created. To export the selected logs, tap the share button on the top right-hand side of the screen.

The iOS Crash Log

If the iOS app isn't responding or is crashing, iOS saves a crash log on the device. You can find the crash log on your device under **Settings › Privacy › Analytics › Analytics Data**. The log entries are sorted alphabetically with the app name, the date and a computer-readable timestamp. Tap the log of choice to open the crash log and export it with the share button on the top right-hand side of the screen.

Capturing App Debug Logs

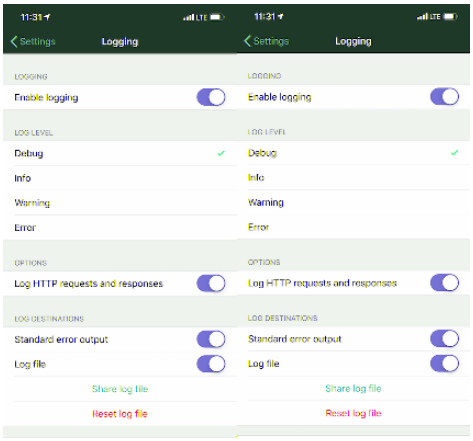
Effectively debugging software requires as much relevant information as possible. Log output can help with tracking down problems and, if you report a bug, log output can help to resolve an issue more quickly. To assist the ownCloud support personnel, please try to provide as many relevant logs as possible. You can do this by enabling and fully configuring the iOS app's logging functionality via:

- Enabling logging with **Settings › Logging › Enable Logging**
- Setting the log level to "*Debug*" or "*Info*"
- Enable:
  - Log HTTP requests and responses
  - Standard error output
  - Log file

Once these have been enabled:

- Click **[Reset log file]**
- Perform the steps to reproduce the error
- Go back to the Logging settings and click **[Share log file]**

All iOS App logging settings enabled and set

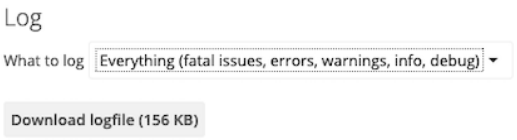


ownCloud's Log File

ownCloud server maintains an ownCloud-specific log file. You can view the file using either the web interface or you can open it directly from the file system in your ownCloud server's data directory.

You can check if it is enabled through the Log configuration panel, which is available under **Settings › General (Admin)**. On that page, you can adjust the log level.
We recommend that you set it to a verbose level such as either `debug` or `info`.

Configuring logging in ownCloud server.



Web Server Log Files

It can be helpful to view your web server's error log file to isolate any ownCloud-related problems.

The ownCloud iOS app sends the `X-REQUEST-ID` header with every request. You'll find the `X-REQUEST-ID` in the `owncloud.log`, and you can configure your webserver to add the `X-REQUEST-ID` to the logs. Here you can find more information at
Request Tracing

Some helpful files include the following:

error_logx

Maintains errors associated with PHP code.

access_log

Typically records all requests handled by the server; handy as a debugging tool, because the log line contains information specific to each request and its result.

Below, you can find where the error logs are typically located, based on operating system and web server.

Operating System	Web Server	File Location
Linux	Apache	`/var/log/apache2`
	NGINX	`/var/log/nginx`
	Lighttpd	`/var/log/lighttpd`
Windows	Apache	The Windows Event Log or in the `logs` directory relative to the Apache installation directory.
	NGINX	Commonly in the `logs` directory relative to the NGINX installation directory.

You can always check your web server's configuration to know where the log files are located.

Recording the Screen

In iOS 11 or later, you can create a screen recording to better illustrate an error.
If you are not familiar with creating one, follow these instructions.

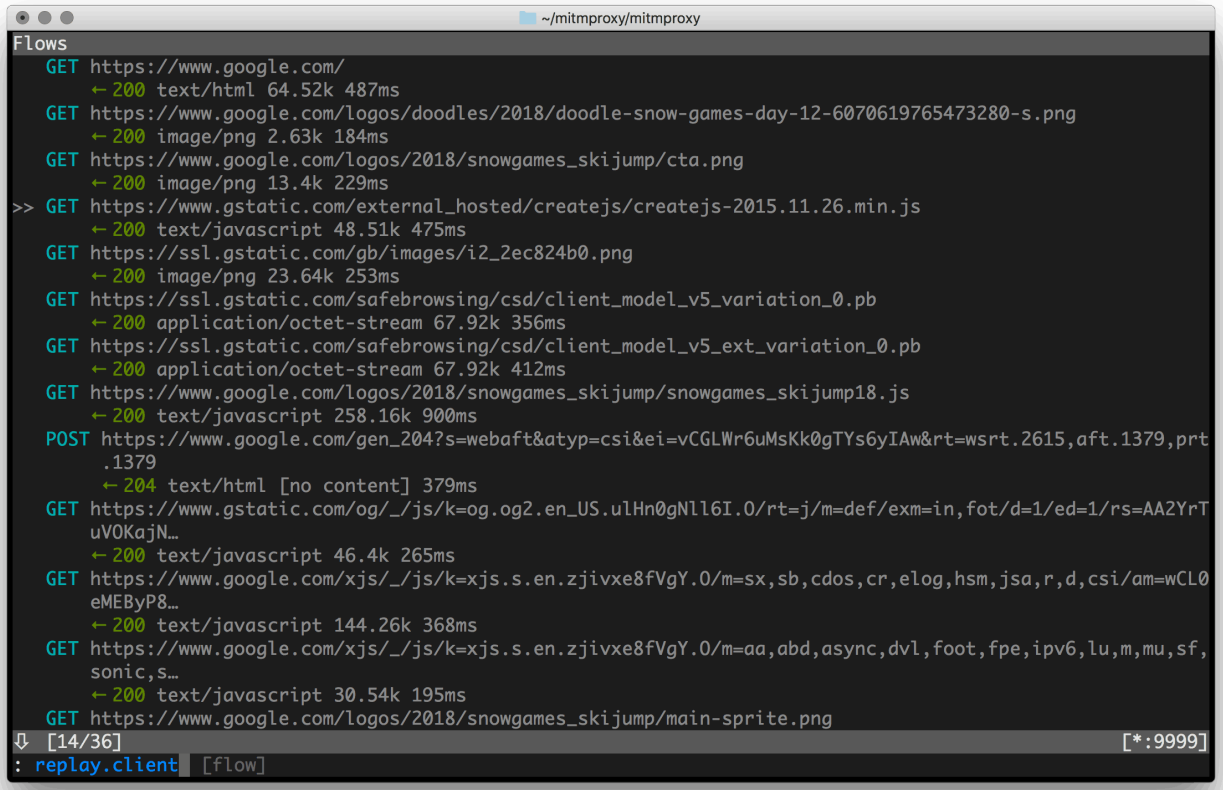
Debugging Tools

If you need to check the traffic between ownCloud and the iOS App, we recommend two tools:

- mitmproxy
- Charles for iOS

mitmproxy

mitmproxy is an interactive man-in-the-middle proxy for HTTP and HTTPS with a console interface. At ownCloud, we use it a lot to investigate every detail of HTTP requests and responses.



Charles for iOS

The Charles proxy for iOS works similarly to mitmproxy. However, it's more user-friendly, runs on the iOS device, *and* has a beautiful UI. It also supports split view on iPads so that you can work with the ownCloud iOS app and Charles side-by-side.

Release Notes

Changelog for the iOS App

ownCloud provides a full changelog with a summary and details for each release of the iOS App. Click the following link to access it at GitHub.