iCloud Storage

Richard Warren

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Who Am I?

Software Engineer:

- Background in DoD funded R&D
- Now focused on iOS apps

Technical Writer:

- Creating iOS 5 Apps
- Objective-C Bootcamp
- Many MacTech Articles

Technical Trainer:

- Senior trainer at About Objects
- Also taught classes for Future Media Concepts and MacAmerica

FMS Labs

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Word of Warning:

Most of my experience is iOS specific

I will try to mention any OS X issues that I am aware of.
You have been warned.

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Last Year

I Was Wrong

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Out of 135 Apps
10 use iCloud Document Storage

5 of those are Apple Apps

3 are Games

Only 2 are apps where I actively use iCloud

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Syncing is Hard

...but some parts are harder than others

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Why Is iCloud Hard?

- Considerably complex than most Apple frameworks
- Initially suffered from poor documentation and a general lack of sample code
- Many of the procedures are opaque
- No obvious debugging tools or techniques
- Often works on a longer time scale

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Litany of Problems

- Users may modify, move or delete a file on one device while viewing it on another.
- Users may log out of their iCloud account, log into another account or delete all their data.
- Users may not even have an active iCloud account, or may not enable it for data storage.
- Users may make conflicting changes on two different devices.
- Users might delete and reinstall the app.
- Users may upgrade the app on one device, but not on another.

Life would be so much easier if we could just get rid of our users.

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

How Does iCloud Work?

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Two Main APIs

- Key Value Storage
- Document Storage
  - Core Data Document storage

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Key Value Storage

- Relatively simple API
- Very similar to NSUserDefaults
- Allows storing plist data types
- You can use it, even if the user doesn't have an active iCloud account

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

KVS Performance

- Up to 1024 keys
- Up to 1 MB of storage
  - This does not come out of the user's iCloud storage.
- Up to 15 request every 90 seconds before throttling
- Last change wins

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Steps to Implement

- Request KVS support in the apps entitlements
- Listen for change notifications
- Get and set values to the NSUbiquitousKeyValueStore

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Four Types of Notifications

- NSUbiquitousKeyValueStoreServerChange
- NSUbiquitousKeyValueStoreInitialSyncChange
- NSUbiquitousKeyValueStoreQuotaViolationChange
- NSUbiquitousKeyValueStoreAccountChange

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Paired With NSUserDefaults

- NSUserDefaults holds the true value
- Whenever we set a value to NSUserDefaults, set KVS as well
- When we get updates from KVS, examine the incoming value and determine if we should update our NSUserDefaults
- If we keep the local value, we should resubmit it to KVS

Document Storage

- A much richer API (also more complex)
- Designed for syncing large amounts of user-generated data
- Anything that can be saved to a file or a file package (plus Core Data).

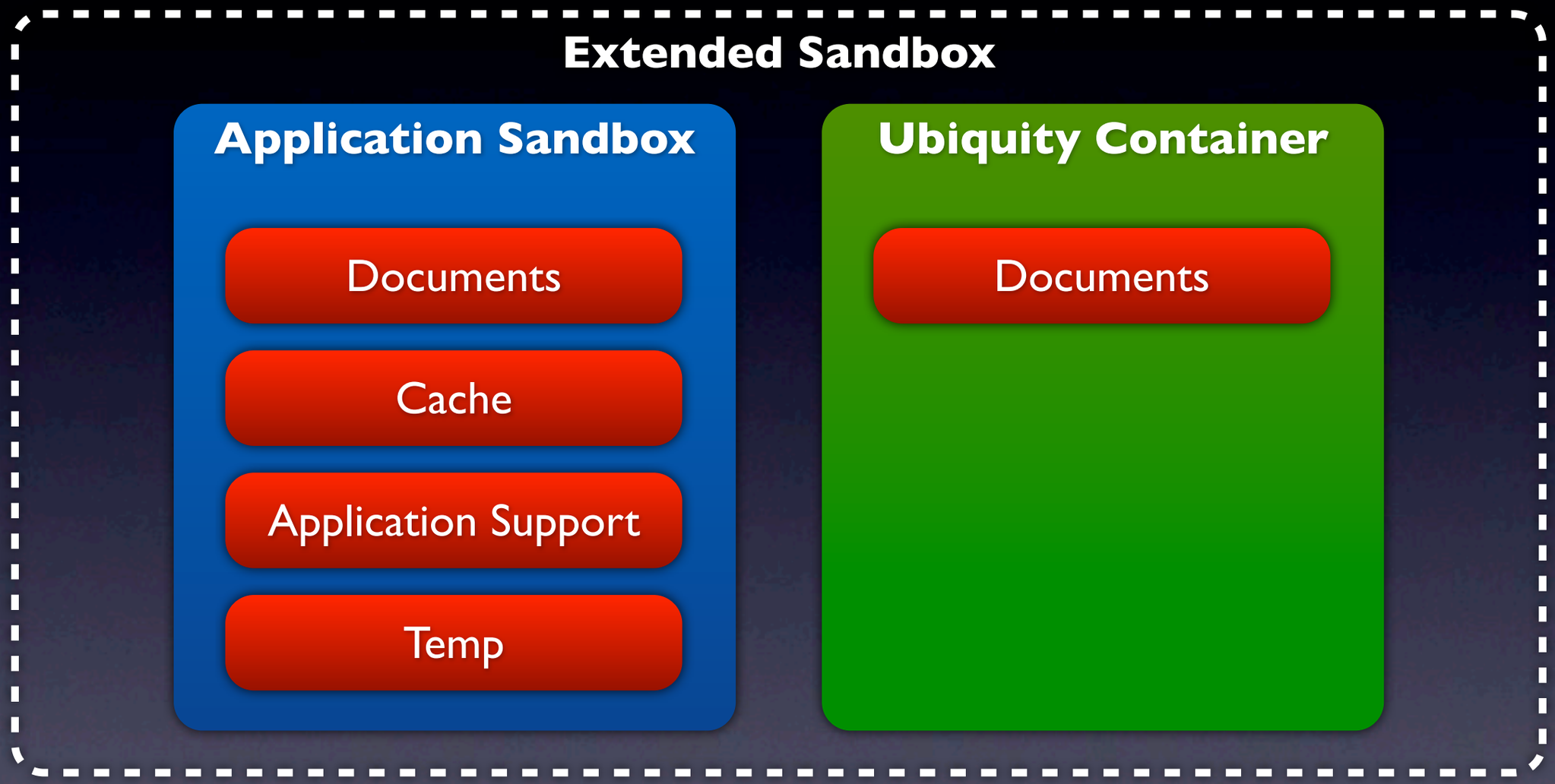
The Big Picture

- Save data into a special location (ubiquity container)
- The system monitors this container, when it sees a change, it uploads the data to the cloud
- Our devices download these changes
- Our app responds to these changes.

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

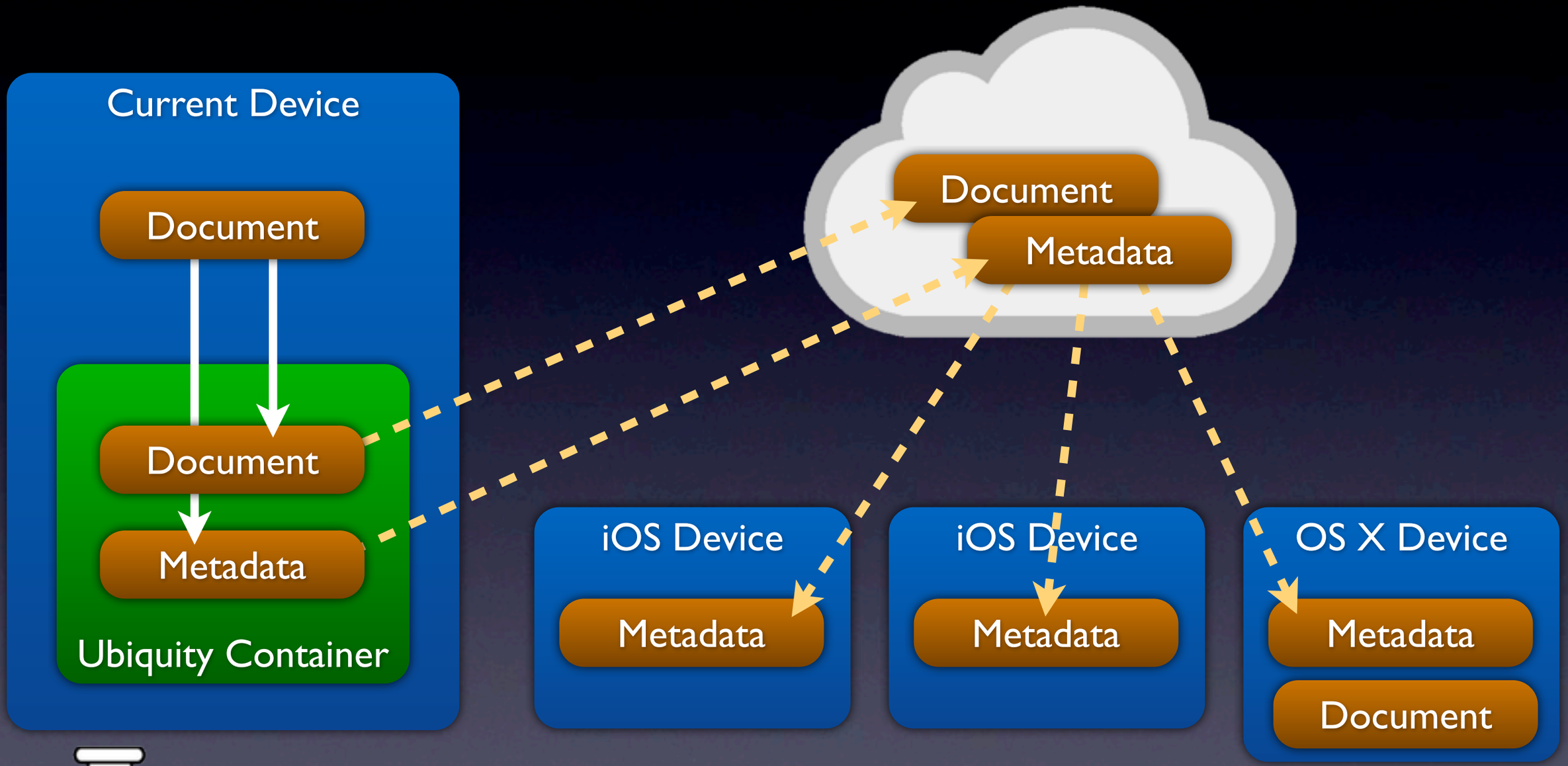
iOS File System



Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Syncing Data



Advantages

- ✓ Work offline
- ✓ Minimizes bandwidth usage
- ✓ All devices notified of changes
- ✓ Devices download data on demand

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Common Strategies

- Use iCloud when available, otherwise fall back to the sandbox
- Ask the user the first time iCloud becomes available
- Let the user choose for each file
- How do users change this decision, and what happens to their documents.

Typical Tasks

1. Setup the application's entitlements
2. Check and monitor the current iCloud account
3. Initialize the ubiquity container
4. Search for existing documents in iCloud using NSMetadataQuery
5. Create new documents and move them into iCloud
6. Coordinate all read/write operations using an NSFilePresenter and NSCoordinator
7. Identify and resolve document conflicts
8. Listen and handle updates and other document state changes

Entitlements

- For Document Storage, we can set an array of ubiquity container identifiers
- We can share a ubiquity container among multiple apps using the same team id.
- The first identifier is the app's primary ubiquity container
- <TEAM_ID>.<BUNDLE_IDENTIFIER>

Monitoring iCloud Account

- Get the current iCloud token:
  id currentToken = [[NSFileManager defaultManager] ubiquityIdentityToken];
- Monitor changes:
  NSUbiquityIdentityDidChangeNotification
- For iOS 5:
  Aggressively call URLForUbiquityContainerIdentifier: whenever the app comes into the foreground

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Initializing the Ubiquity Container

- Call URLForUbiquityContainerIdentifier:
- Must be called on a background thread
- Returns the URL for the specified Ubiquity container
- Extends the apps sandbox, letting the app read and write to the ubiquity container

NSMetadataQuery

- Setup the query, including a predicate
- Listen for notifications
- Search runs in two phases:
  - Initial phase gathers the currently-available metadata
  - Live-update phase continues to monitor the ubiquity container for changes
- OS X, automatically manages this through NSDocument's Open and Save dialogs

Coordinate All File IO

- Files in the ubiquity container may be read and modified by two processes:
  - Our App
  - The iCloud Service
- We coordinate and monitor changes using NSFileCoordinator and NSFilePresenter

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

NSFilePresenter

- A protocol we can implement for low-level monitoring of files or directories
- Receives notifications whenever a coordinated write modifies or moves the file
- May be asked to save any current changes
- May be asked to accommodate deletions
- May be asked to relinquish control to coordinated readers/writers

NSFileCoordinator

- Block Based API
- Allows us to coordinate reading and writing
  - Allows any number of read actions
  - Only one write action at a time
  - Write actions block until all the current read actions end
  - Both read and write actions block until the current write action ends.
- in iOS coordinated reads may initiate file downloads

Uploading and Downloading

- Usually occurs automatically
- We can check the file's state using NSURL's getResourceValue:forKey:error:
- Can manually trigger downloads or remove downloaded files using NSFileManager methods
  - startDownloadingUbiquitousItemAtURL:error:
  - evictUbiquitousItemAtURL:error:

UIDocument

- Enables automatic background saving
- Enables undo support
- Implements the NSFilePresenter protocol
- Automatically creates NSFileCoordinators
- Automatically tracks any changes to the file's location or name
- Provides a framework for detecting and resolving conflicts

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

UIDocument (cont.)

- Abstract Class
- We must implement two methods to read and write data:
  - loadFromContents:ofType:error:
  - contentsForType:error
- Must listen for UIDocumentStateChangedNotification

iCloud and Core Data

- Combines two very complex technologies
- What could possibly go wrong...

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

SQLite Databases

- Cannot place an SQLite Database into iCloud
- Instead, iCloud's Core Data support uses local SQLite databases and syncs the transaction logs
- We can rebuild any database by simply reading in the transaction logs
- In many ways, the persistent store is a local cache of the database

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Adding iCloud Support

- We just need to set two properties to start syncing:
  - NSPersistentStoreUbiquitousContentNameKey
  - NSPersistentStoreUbiquitousContentURLKey
- Then listen for updates:
  - NSPersistentStoreDidImportUbiquitousContentChangesNotification.
- If we delete persistent store, we must delete the transaction logs

Importing Changes

```
- (void)iCloudUpdate:(NSNotification *)note {
 [self.moc performBlock:^{
     [self.moc mergeChangesFromContextDidSaveNotification:note];
 }];
}
```

```
// updating a fetched results controller
[[self fetchedResultsController] performFetch:&error];

// updating individual objects
NSManagedObjectID *personID = [person objectID];
self.person = (Person *)[self.moc objectWithID:personID];
```

Store Locations

- In the Application Sandbox
  - Let's the user continue to use the data when logged out of iCloud
  - Easy to get into an inconsistent state
- In the Ubiquity Container
  - Use .nosync to prevent syncing
  - The data won't be available when the user logs out

Other Implications

- Our app uses two different URLs for each document:
  - the location of the persistent store
  - the location of the transaction logs
- We cannot place a database with existing data into iCloud
- If the transaction logs and database get out of sync, iCloud syncing stops working

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

UIManagedDocument

- Concrete UIDocument class
- Provides a parent/child hierarchical MOC
- Automatically builds the Core Data stack
- Saves the persistent store in a .nosync directory
- Automatic three-way conflict resolution
- NSPersistentDocument does not support iCloud syncing

Recommendations

- Use UIDocument, UIManagedDocument or NSDocument, if possible
- Make sure the document is working in the sandbox before implementing iCloud
- Override methods to provide better logging:
  - writeContents:toURL:forSaveOperation:originalContentsURL:error:
  - handleError:userInteractionPermitted:
- Start with a simple iCloud strategy

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

More Recommendations

- Avoid syncing device specific information (e.g. scroll location)
- Avoid creating sync storms (last modified timestamps)
- Use case-sensitive file names
- Watch out for device-specific data types
- Always version your data
- Report bugs

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Test iCloud in Steps

1. Make sure your app always autosaves before going into the background
2. Make sure the data is getting to the cloud
3. Make sure it is syncing
4. Use airplane mode to create conflicts
5. Be sure to test logouts, deleting data and deleting and reinstalling apps.

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Examining iCloud Data

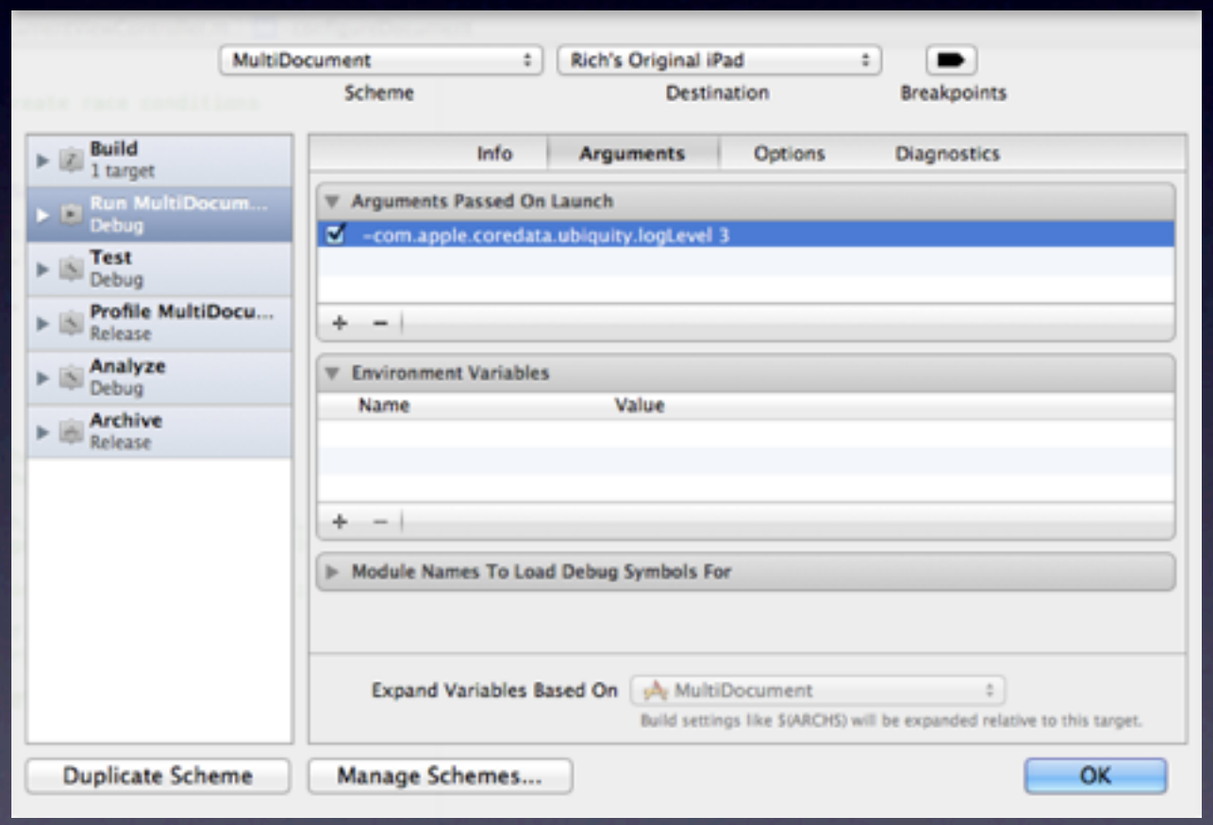
- Settings and System Preferences
- ~/Library/Mobile Documents
- https://developer.icloud.com
- For Core Data, use an NSMetadataQuery to track incoming transaction logs

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

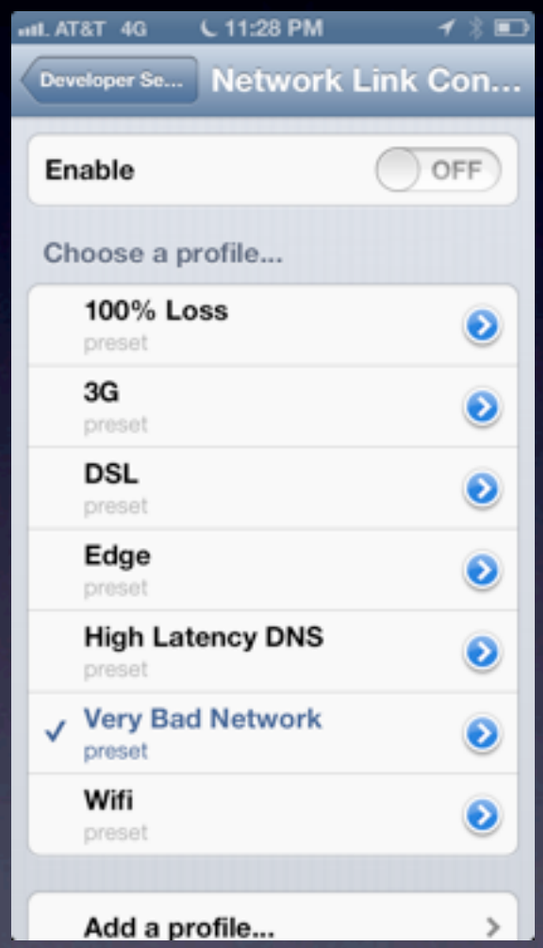
Enable Debug Log

com.apple.coredata.ubiquity.loglevel 1, 2 or 3



iOS Developer Settings

Settings > Developer > Network Link Conditioner



Return To A Clean Slate

- Delete the data from iCloud
- Wait for the delete to propagate to all the devices
- Uninstall the apps
- Reinstall and rerun
- Occasionally you may need to programmatically delete everything in the ubiquity container

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

iCloud vs Dropbox

- Dropbox is platform agnostic
- It has a simpler API
- It doesn't offer all of iCloud's features
- It requires users to sign up for a 3rd party service

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

The Future

- Apple is betting big on iCloud
  - Documentation and sample code will continue to improve
  - API will also improve
  - Hopefully debugging and troubleshooting tools will improve as well.
- We are moving towards a future where all our devices are windows onto a common set of data

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Resources

- Creating iOS 5 Apps
- Adopting iCloud Storage Videos
- WWDC 2012 Videos
- iCloud Design Guide

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting

Richard Warren
rich@freelancemadscience.com
http://freelancemadscience.com
@rikiwarren
+Rich Warren

Questions?

Freelance Mad Science Labs

iOS Development, Technical Writing, Training and Consulting