Using iCloud with Core Data

Session 227

Adam Swift
Senior Software Engineer

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

Introduction

- Take app from idea to iCloud
- Beyond the API
- Real-world strategies

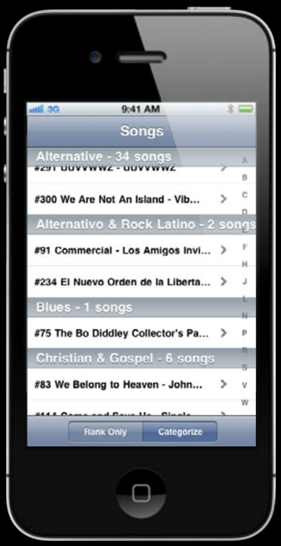
[redacted]

Sample Code

https://developer.apple.com/wwdc/schedule/details.php?id=227

Before We Begin…

Core Data



Alternative - 34 songs

#291 UUVVWWZ - UUVVWWZ

#300 We Are Not An Island - Vib...

Alternativo & Rock Latino - 2 songs

#91 Commercial - Los Amigos Invi...

#234 El Nuevo Orden de la Liberta...

Blues - 1 songs

#75 The Bo Diddley Collector's Pa...

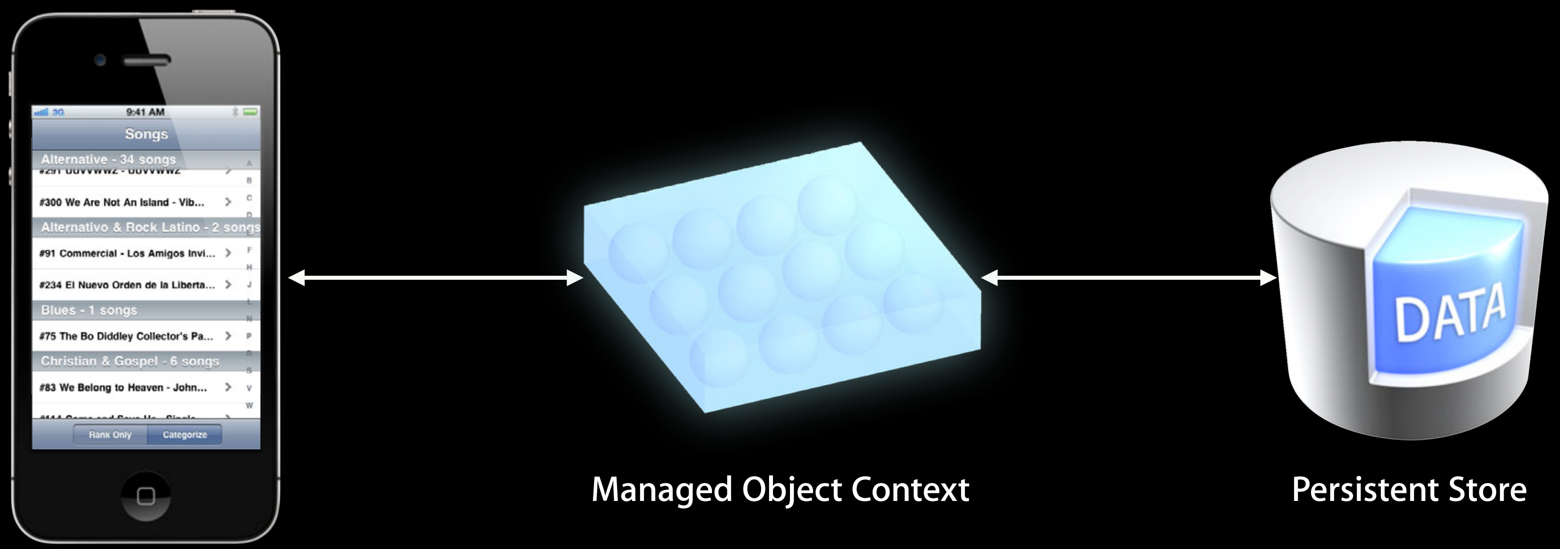
Christian & Gospel - 6 songs

#83 We Belong to Heaven - John...

#114 Come and Save Us - Single

Persistent Store

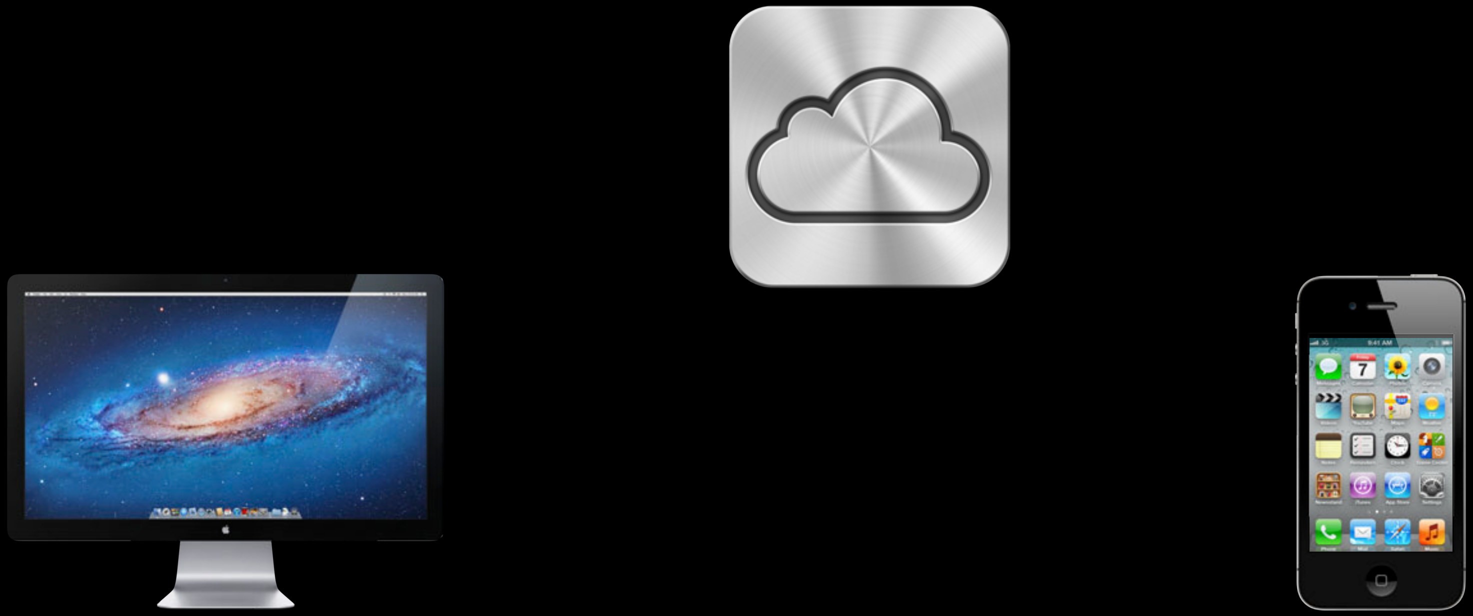
Core Data


9:41 AM
Songs
Alternative - 34 songs
#300 We Are Not An Island - Vib...
Alternativo & Rock Latino - 2 songs
#91 Commercial - Los Amigos Invi...
#234 El Nuevo Orden de la Liberta...
Blues - 1 songs
#75 The Bo Diddley Collector's Pa...
Christian & Gospel - 6 songs
#83 We Belong to Heaven - John...
Rank Only
Categorize
DATA

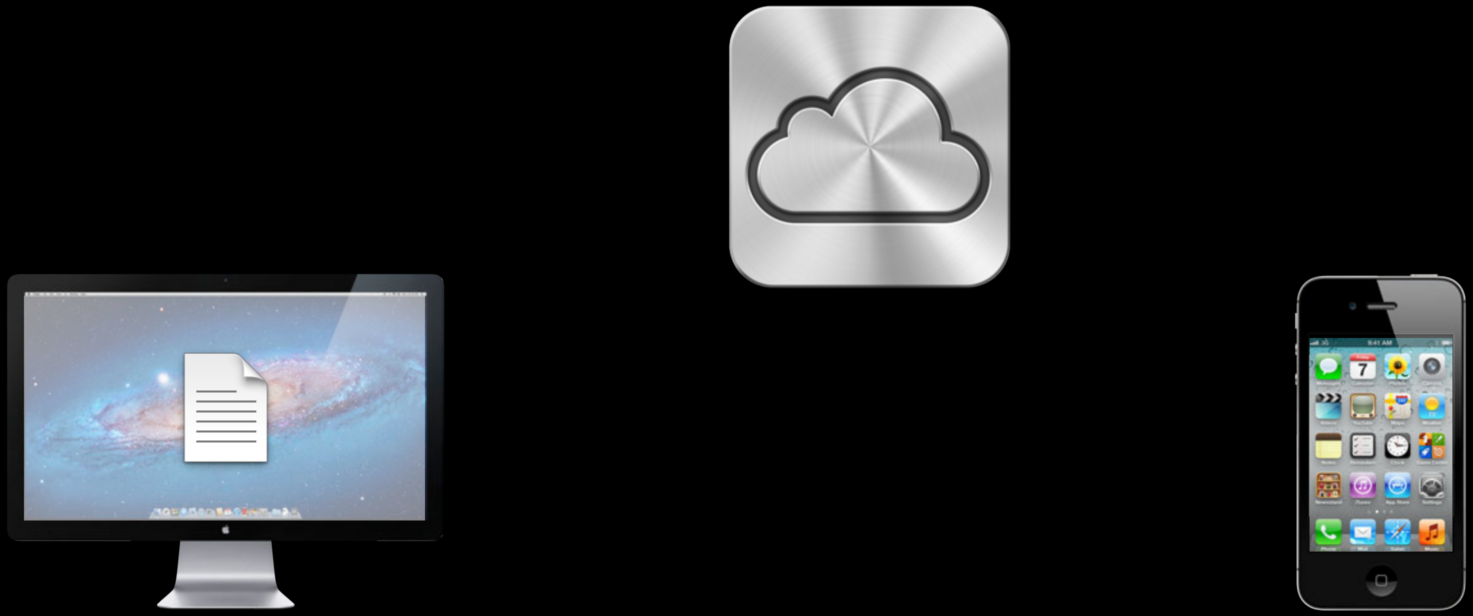
Managed Object Context

Persistent Store

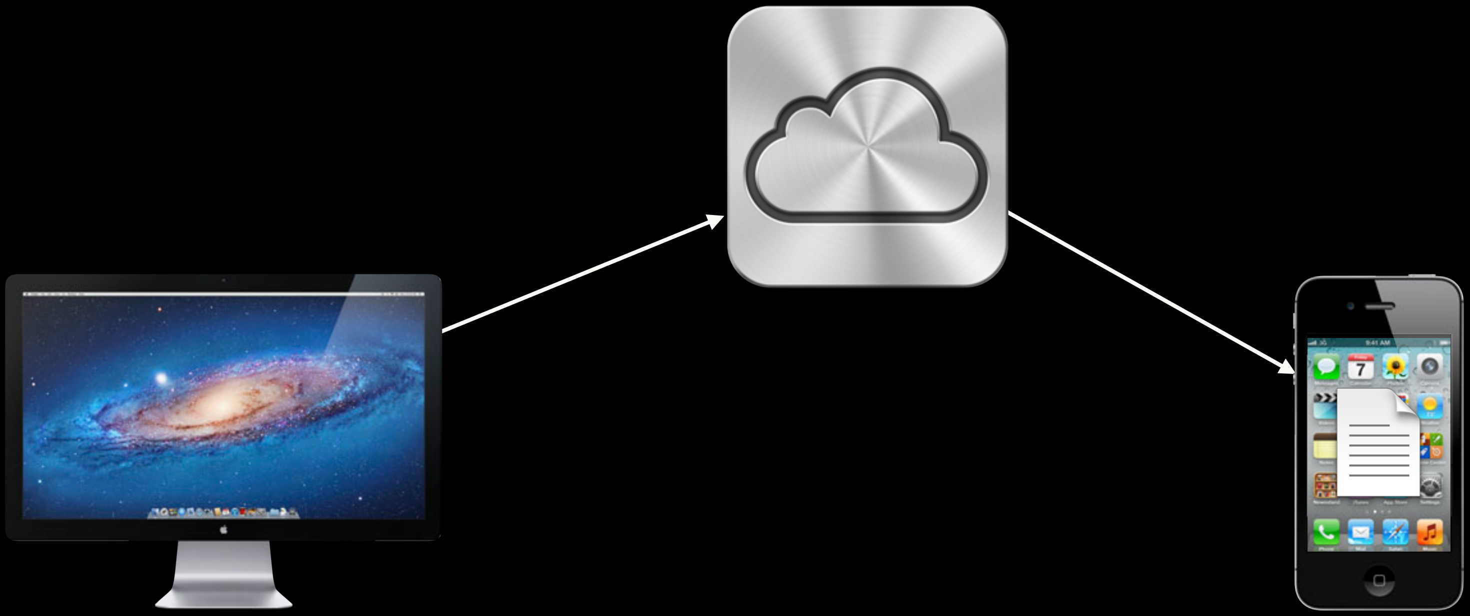
Core Data + iCloud



Core Data + iCloud



Core Data + iCloud



Core Data iCloud Features

- Per record conflict resolution
- Three-way merge

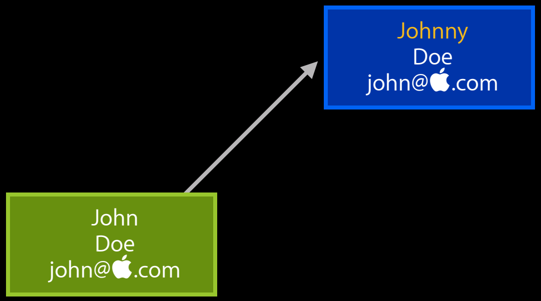
Three-Way Merge

Preserve changes between systems

John
Doe
john@.com

Three-Way Merge

Preserve changes between systems

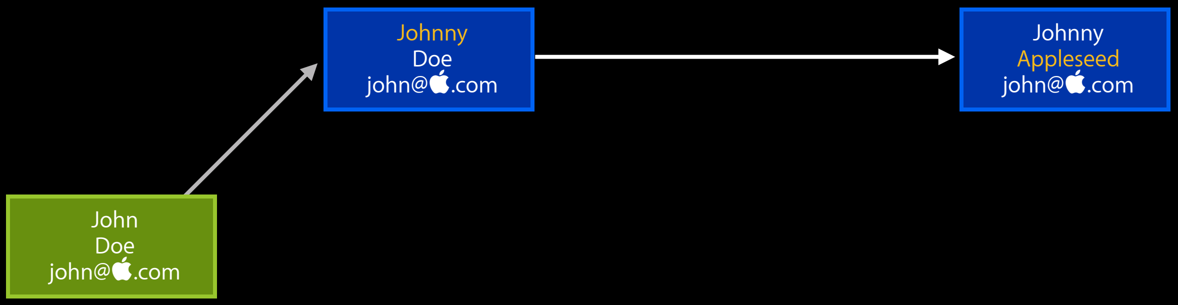


Johnny
Doe
john@.com

John
Doe
john@.com

Three-Way Merge

Preserve changes between systems



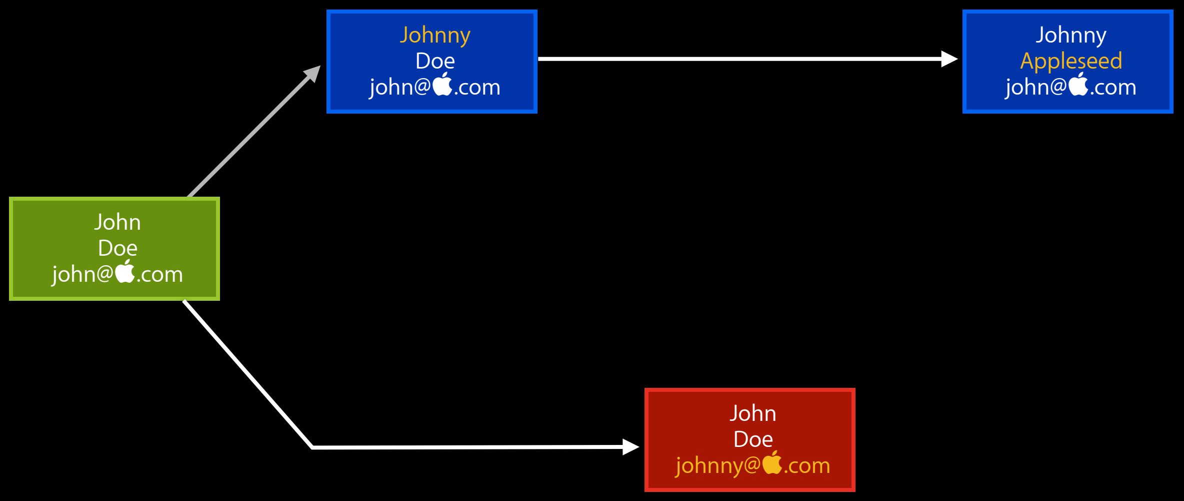
John
Doe
john@.com

Johnny
Doe
john@.com

Johnny
Appleseed
john@.com

Three-Way Merge

Preserve changes between systems



John
Doe
john@.com

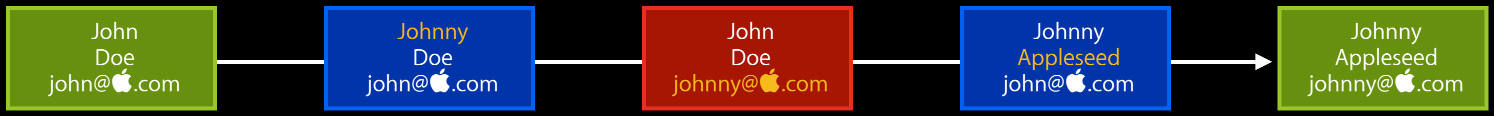
Johnny
Doe
john@.com

Johnny
Appleseed
john@.com

John
Doe
johnny@.com

Three-Way Merge

Preserve changes between systems



John
Doe
john@.com

Johnny
Doe
john@.com

John
Doe
johnny@.com

Johnny
Appleseed
john@.com

Johnny
Appleseed
johnny@.com

Core Data iCloud Features

- Transfer incremental changes
- Asynchronous import
- Lightweight schema migration

What You Need

- Xcode and OS X/iOS SDK
  - Core Data project template (or Sample Code)
  - iCloud Entitlements

What You Need

- Xcode and OS X/iOS SDK
  - Core Data project template (or Sample Code)
  - iCloud Entitlements

- Provisioning Portal
  - App ID for iCloud
  - Provisioning Profile

Design

Launch

Lifecycle

Design

Launch

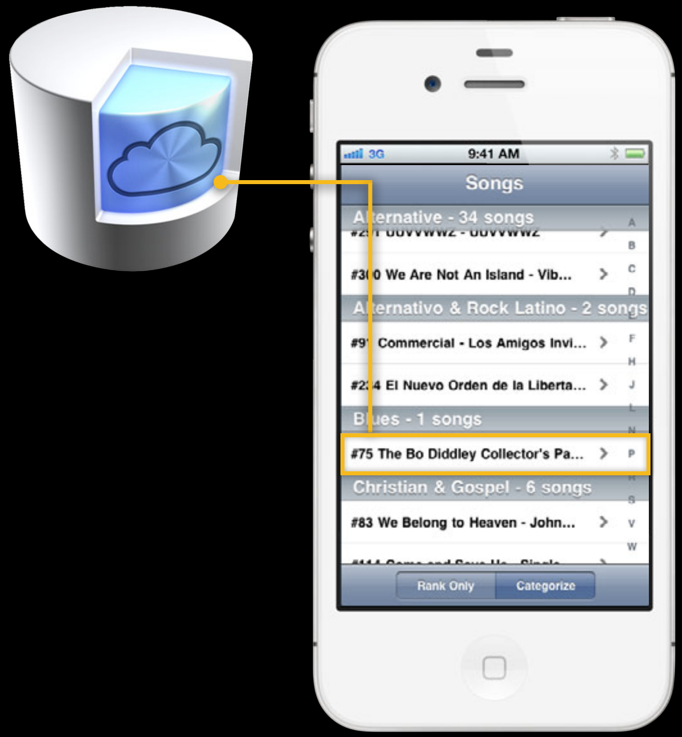
Lifecycle

User Expectations

- What data in iCloud?
- Consider the variables
  - Network
  - Account
  - Data

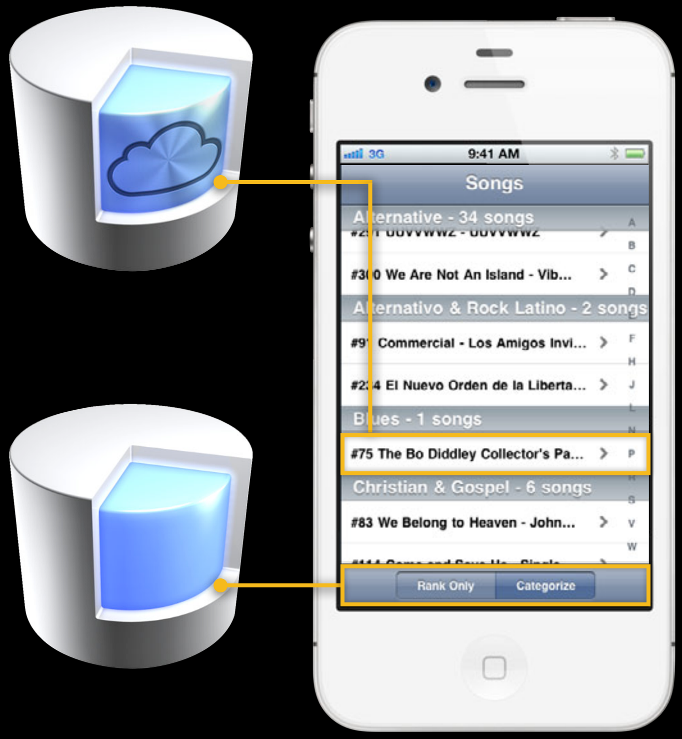
Where Does the Data Go?

- Different types of apps
- Where does app data live?
  - All in iCloud
  - Some in iCloud, some local


3G
9:41 AM
Songs
Alternative - 34 songs
#300 We Are Not An Island - Vib...
Alternativo & Rock Latino - 2 songs
#9 Commercial - Los Amigos Invi...
#214 El Nuevo Orden de la Liberta...
Blues - 1 songs
#75 The Bo Diddley Collector's Pa...
Christian & Gospel - 6 songs
#83 We Belong to Heaven - John...
Rank Only
Categorize

Where Does the Data Go?

- Different types of apps
- Where does app data live?
  - All in iCloud
  - Some in iCloud, some local


3G
9:41 AM
Songs
Alternative - 34 songs
#300 We Are Not An Island - Vib...
Alternativo & Rock Latino - 2 songs
#9 Commercial - Los Amigos Invi...
#224 El Nuevo Orden de la Liberta...
Blues - 1 songs
#75 The Bo Diddley Collector's Pa...
Christian & Gospel - 6 songs
#83 We Belong to Heaven - John...
Rank Only
Categorize

What Goes into iCloud?

- Data goes into iCloud

Persistent Store

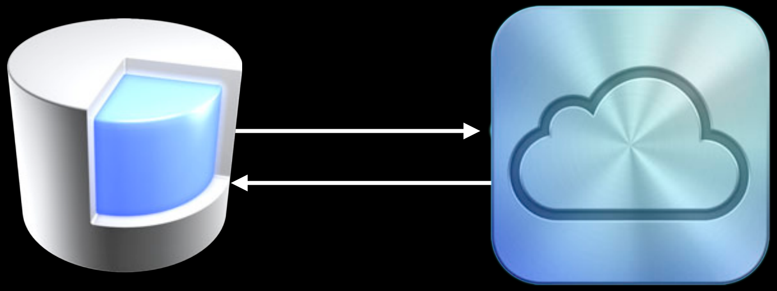
What Goes into iCloud?

- Data goes into iCloud
- Not Persistent Store file!

Never open an
SQLite database in iCloud

What Goes into iCloud?

- Data goes into iCloud
- Not Persistent Store file!
- Data transferred as incremental changes via iCloud



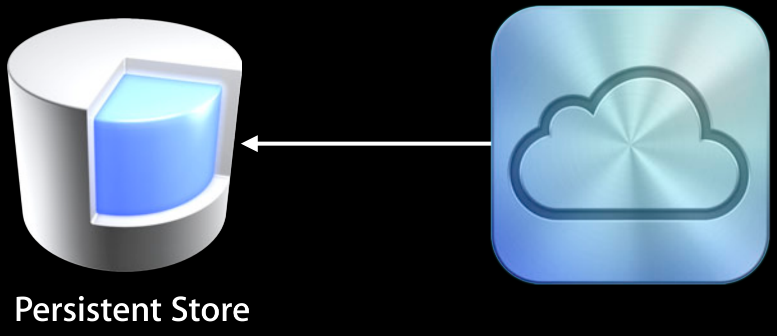
Persistent Store

Persistent Store as Cache

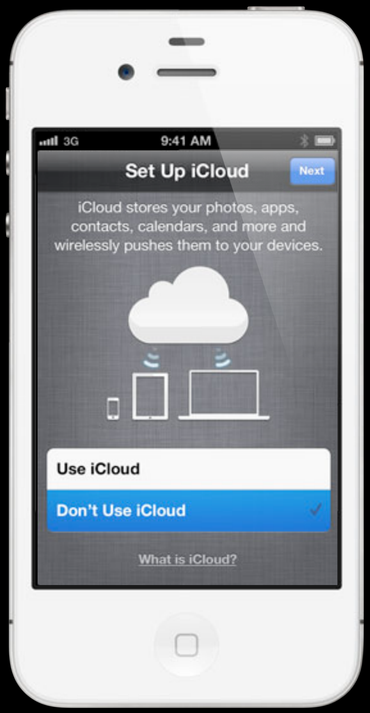
- Data in iCloud
- Access via Persistent Store
- Rebuild from data in iCloud

Persistent Store as Cache

- Data in iCloud
- Access via Persistent Store
- Rebuild from data in iCloud



Fallback Store



Set Up iCloud

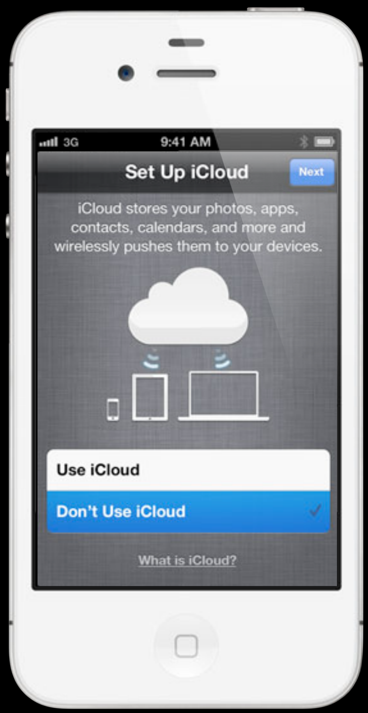
iCloud stores your photos, apps, contacts, calendars, and more and wirelessly pushes them to your devices.

Use iCloud

Don't Use iCloud ✓

Fallback Store

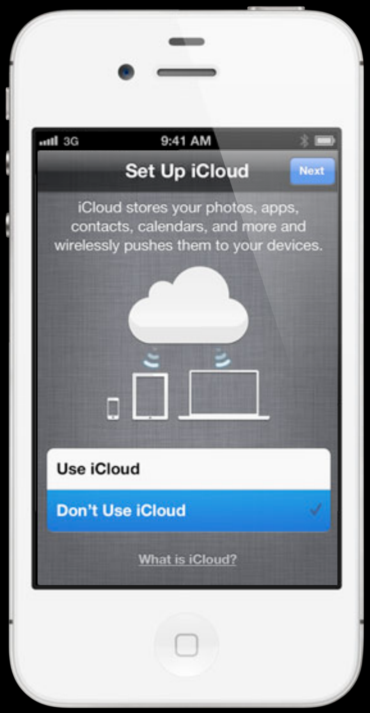
- No iCloud account?
  - Provide seamless app experience
  - Create local ‘fallback’ store



iCloud stores your photos, apps, contacts, calendars, and more and wirelessly pushes them to your devices.

Fallback Store

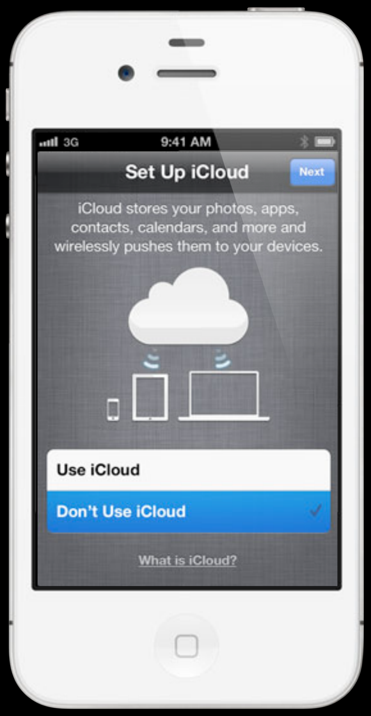
- No iCloud account?
  - Provide seamless app experience
  - Create local 'fallback' store
- Existing store (pre-iCloud app version)



iCloud stores your photos, apps, contacts, calendars, and more and wirelessly pushes them to your devices.

Fallback Store

- No iCloud account?
  - Provide seamless app experience
  - Create local 'fallback' store
- Existing store (pre-iCloud app version)
- Move store data into iCloud when enabled



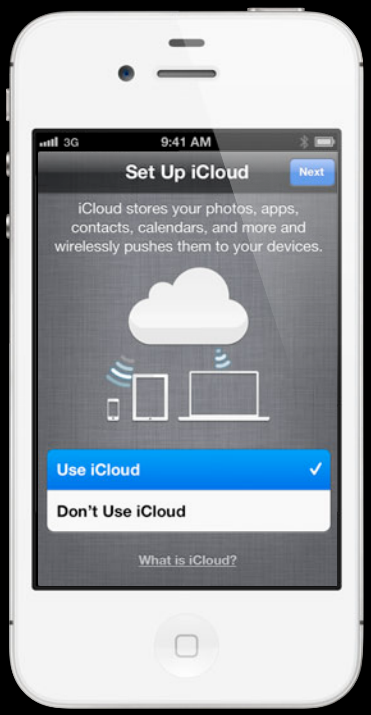
iCloud stores your photos, apps, contacts, calendars, and more and wirelessly pushes them to your devices.

Use iCloud

Don't Use iCloud ✓

iCloud-Enabled Store Location

- Your decision
- Determined by access needs
  - Access requires iCloud account?
  - Or, allow read-only access without account?



Set Up iCloud

iCloud stores your photos, apps, contacts, calendars, and more and wirelessly pushes them to your devices.

Use iCloud ✓

Don't Use iCloud

Put Store in iCloud Container

Requires iCloud account for access

MyFolder.nosync

Put Store in iCloud Container

Requires iCloud account for access

Put Store in iCloud Container

Requires iCloud account for access

Put Store in iCloud Container

Requires iCloud account for access

- In container, but .nosync not transferred to iCloud

Put Store in iCloud Container

Requires iCloud account for access

- In container, but .nosync not transferred to iCloud
- Removed when the account changes
    - Rebuild store from iCloud data

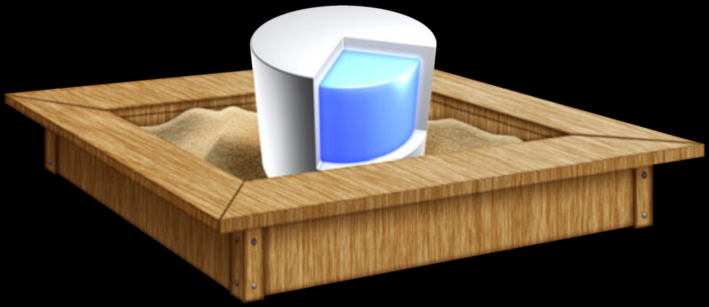
Put Store in iCloud Container

Requires iCloud account for access

- In container, but .nosync not transferred to iCloud
- Removed when the account changes
  - Rebuild store from iCloud data
- Simple
  - Guaranteed to match “ubiquitous content”
  - No iCloud? No store

Keep Store in Sandbox

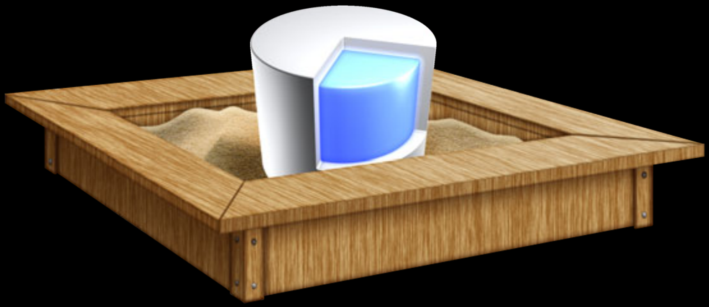
Allows read-only access without account



Keep Store in Sandbox

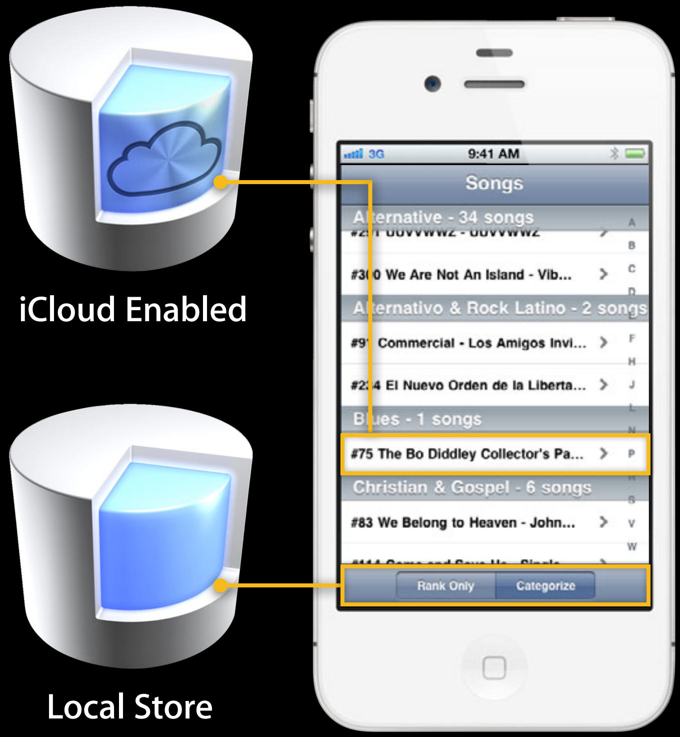
Allows read-only access without account

- Persistent store in App Sandbox, data in iCloud
- Store file survives account changes
- One store per-iCloud account
- Read-only access without account!
  - Use NSReadOnlyPersistentStoreOption
  - Optionally move data to fallback store



Partition Data

- Two Persistent Stores
  - One for local device data
  - One enables data in iCloud
- Use different models or configurations

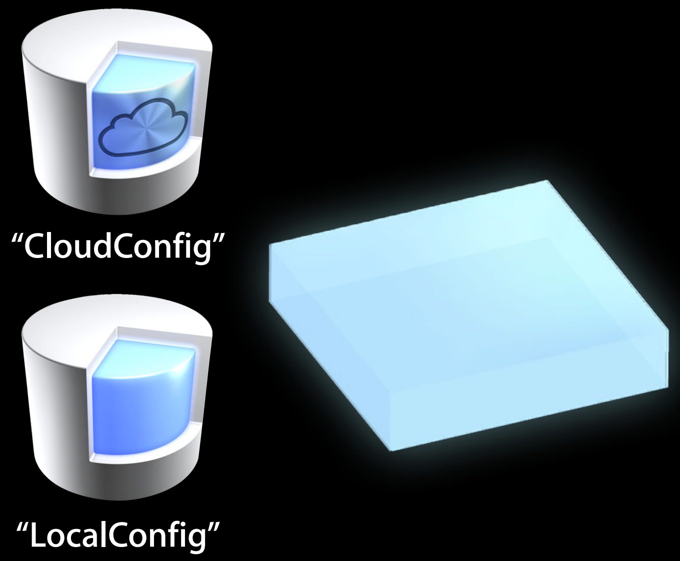

iCloud Enabled

Local Store

3G
9:41 AM
Songs
Alternative - 34 songs
#300 We Are Not An Island - Vib...
Alternativo & Rock Latino - 2 songs
#91 Commercial - Los Amigos Invi...
#234 El Nuevo Orden de la Liberta...
Blues - 1 songs
#75 The Bo Diddley Collector's Pa...
Christian & Gospel - 6 songs
#83 We Belong to Heaven - John...
Rank Only
Categorize

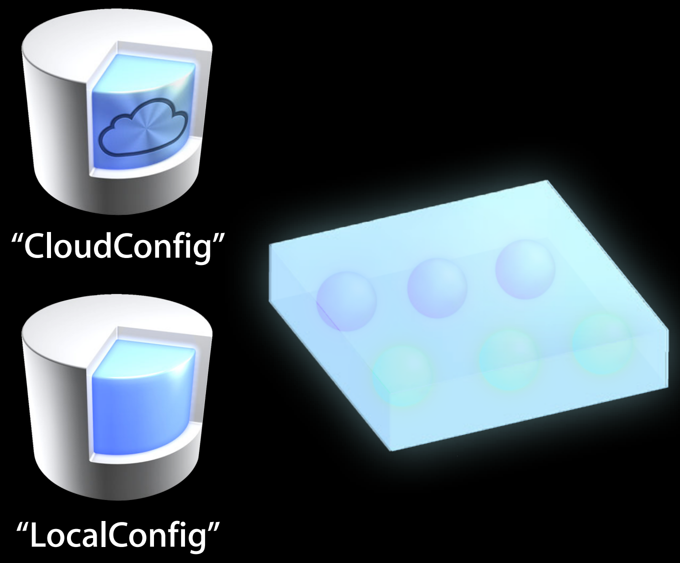
Model Configurations

- Subset of entities from a data model
- Create store with configuration
  - Includes only those entities
- One context can access stores with different configurations (based on the same model)

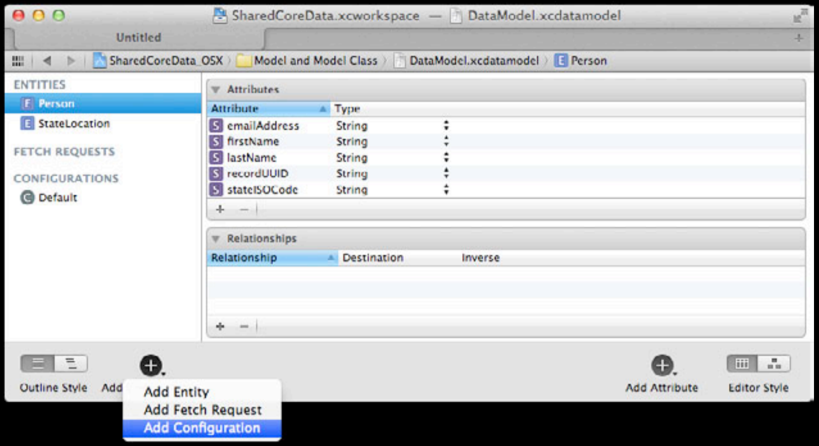


Model Configurations

- Subset of entities from a data model
- Create store with configuration
  - Includes only those entities
- One context can access stores with different configurations (based on the same model)



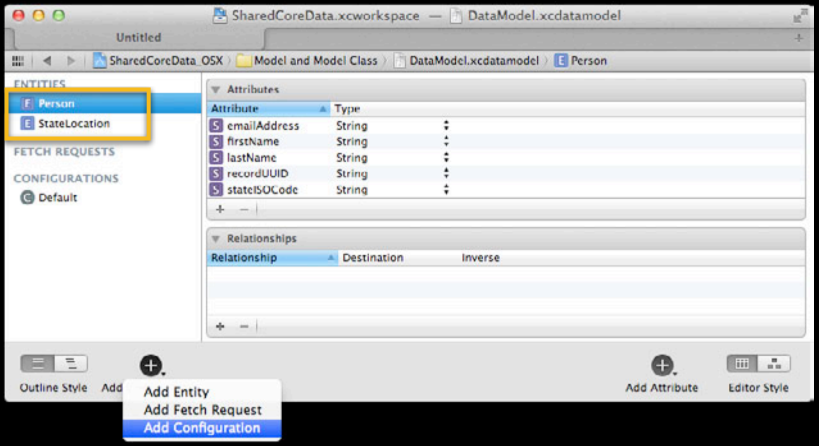
Configurations in Data Model



Attribute	Type
emailAddress	String
firstName	String
lastName	String
recordUUID	String
stateISOCode	String

Relationship	Destination	Inverse

Configurations in Data Model

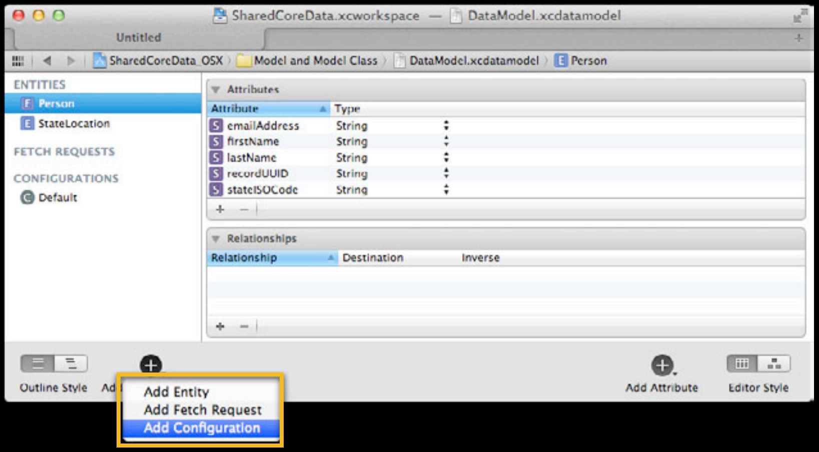


SharedCoreData.xcworkspace — DataModel.xcdatamodel

Attribute	Type
emailAddress	String
firstName	String
lastName	String
recordUUID	String
stateISOCode	String

Relationship	Destination	Inverse

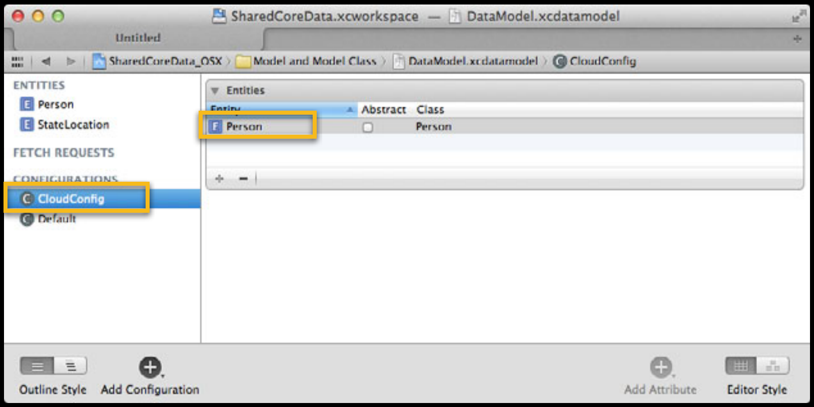
Configurations in Data Model



Attribute	Type
emailAddress	String
firstName	String
lastName	String
recordUUID	String
stateISOCode	String

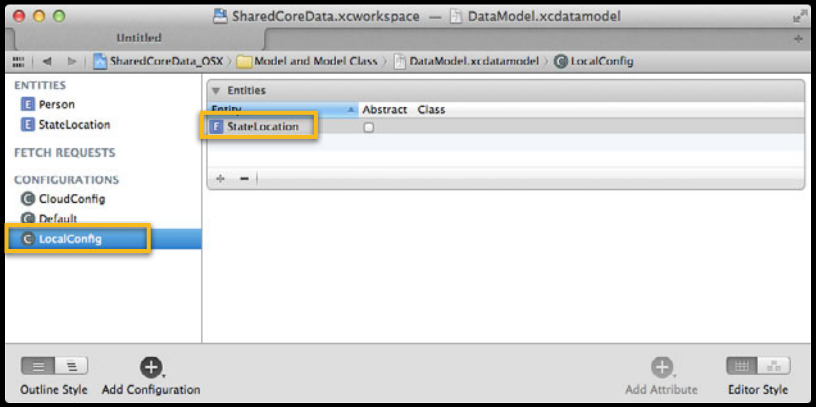
Relationship	Destination	Inverse

Configurations in Data Model



Entity	Abstract	Class
Person	□	Person

Configurations in Data Model



Configurations in Code

```
cloudStore = [psc addPersistentStoreWithType: NSSQLiteStoreType
                               configuration: @”CloudConfig”
                                         URL: cloudStoreURL
                                     options: cloudOptions
                                       error: &error];
```

Configurations in Code

```
cloudStore = [psc addPersistentStoreWithType: NSSQLiteStoreType
                               configuration: @”CloudConfig”
                                         URL: cloudStoreURL
                                     options: cloudOptions
                                       error: &error];
```

Configurations in Code

```
cloudStore = [psc addPersistentStoreWithType: NSSQLiteStoreType
                               configuration: @”CloudConfig”
                                         URL: cloudStoreURL
                                     options: cloudOptions
                                       error: &error];
localStore = [psc addPersistentStoreWithType: NSSQLiteStoreType
                               configuration: @”LocalConfig”
                                         URL: localStoreURL
                                     options: localOptions
                                       error: &error];
```

Summary

- User expectations
- Where data goes
- Store location trade-offs

Design

Launch

Lifecycle

Design

Launch

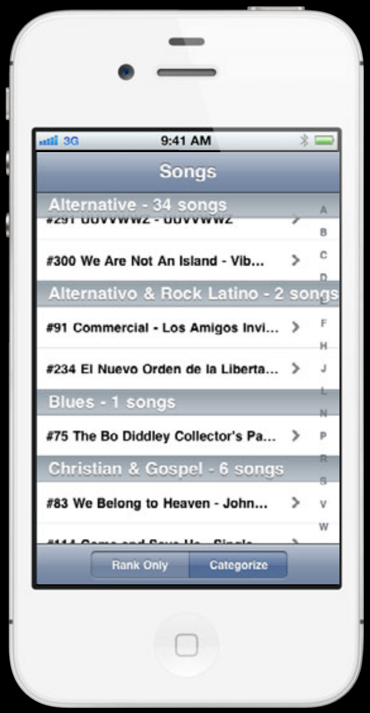
Lifecycle

Launch Steps

- First launch?
- New device, existing data

Goals

- Get to full UI
- Evaluate status
  - Ready to go?
  - Seed?
  - Fallback?
- Stay responsive



Alternative - 34 songs

#291 UUVVWWZ - UUVVWWZ

#300 We Are Not An Island - Vib...

Alternativo & Rock Latino - 2 songs

#91 Commercial - Los Amigos Invi...

#234 El Nuevo Orden de la Liberta...

Blues - 1 songs

#75 The Bo Diddley Collector's Pa...

Christian & Gospel - 6 songs

#83 We Belong to Heaven - John...

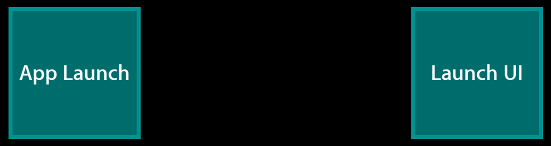
#114 Come and Save Us - Single

- Launch

- Launch

App Launch

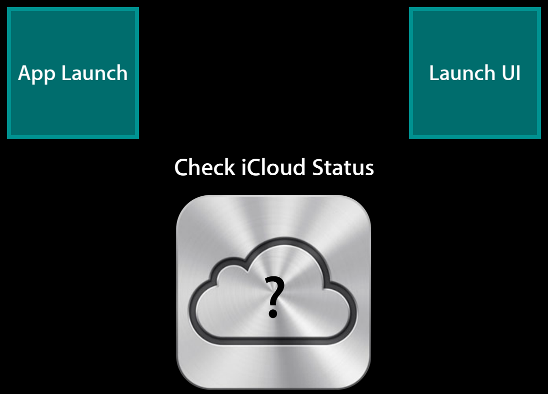
- Launch



App Launch

Launch UI

- Launch

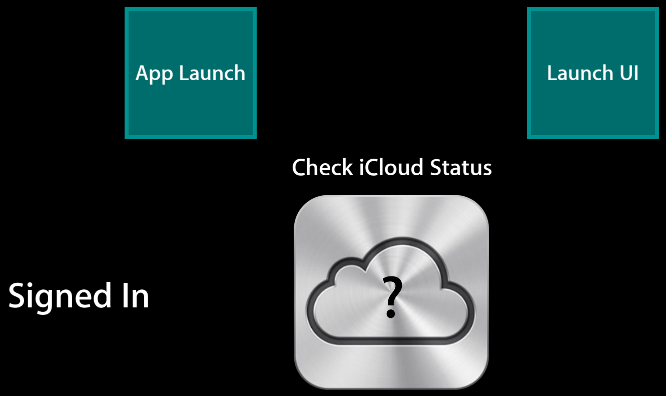


App Launch

Launch UI

Check iCloud Status

- Launch



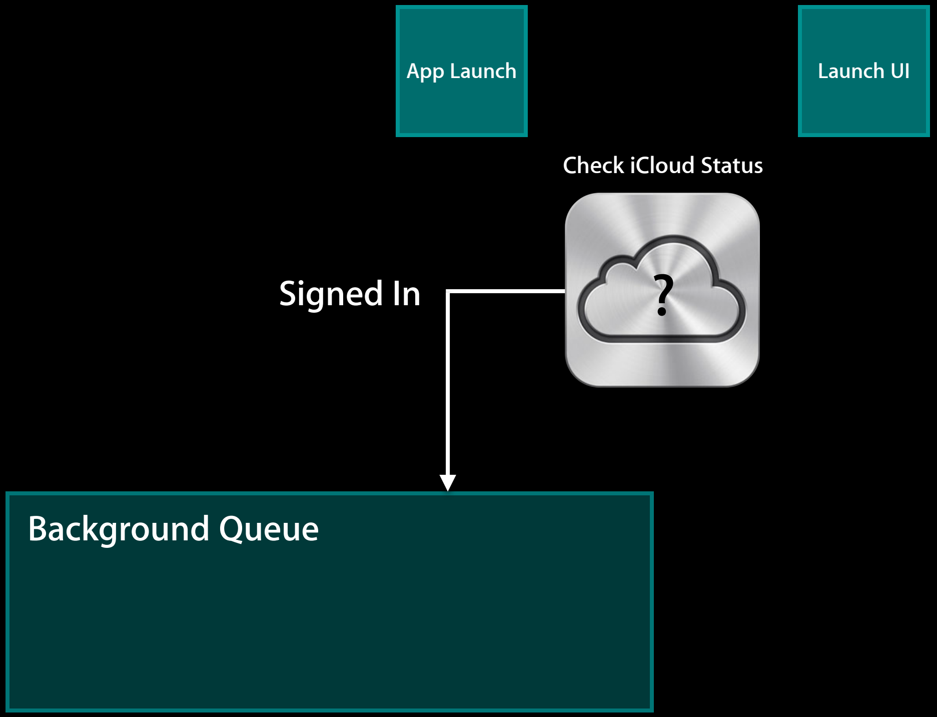
App Launch

Launch UI

Check iCloud Status

Signed In

- Launch



App Launch

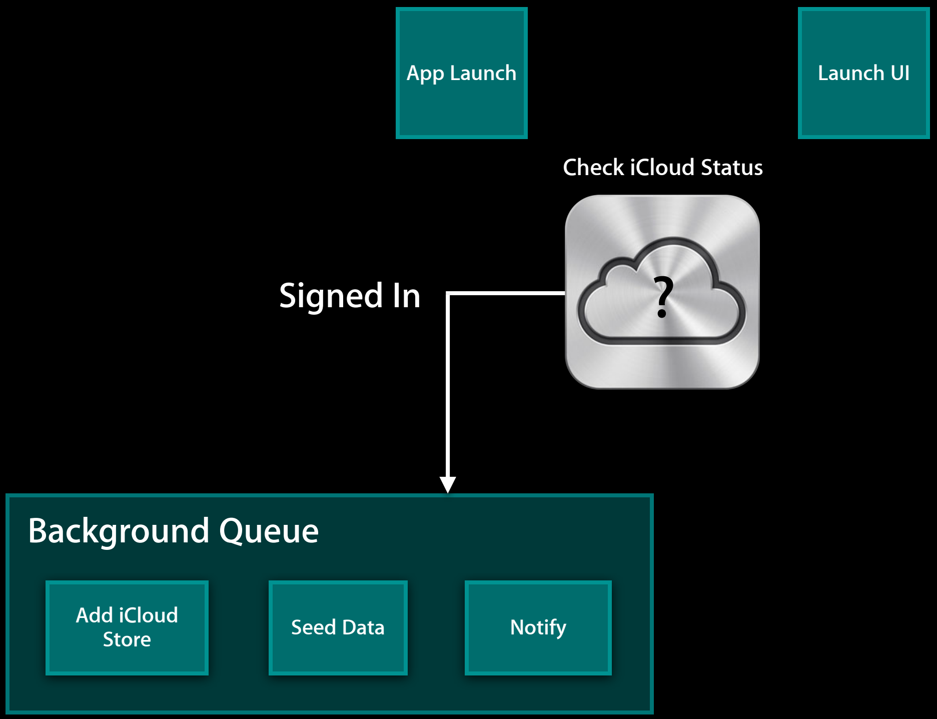
Launch UI

Check iCloud Status

Signed In

Background Queue

- Launch


App Launch

Launch UI

Check iCloud Status

Signed In

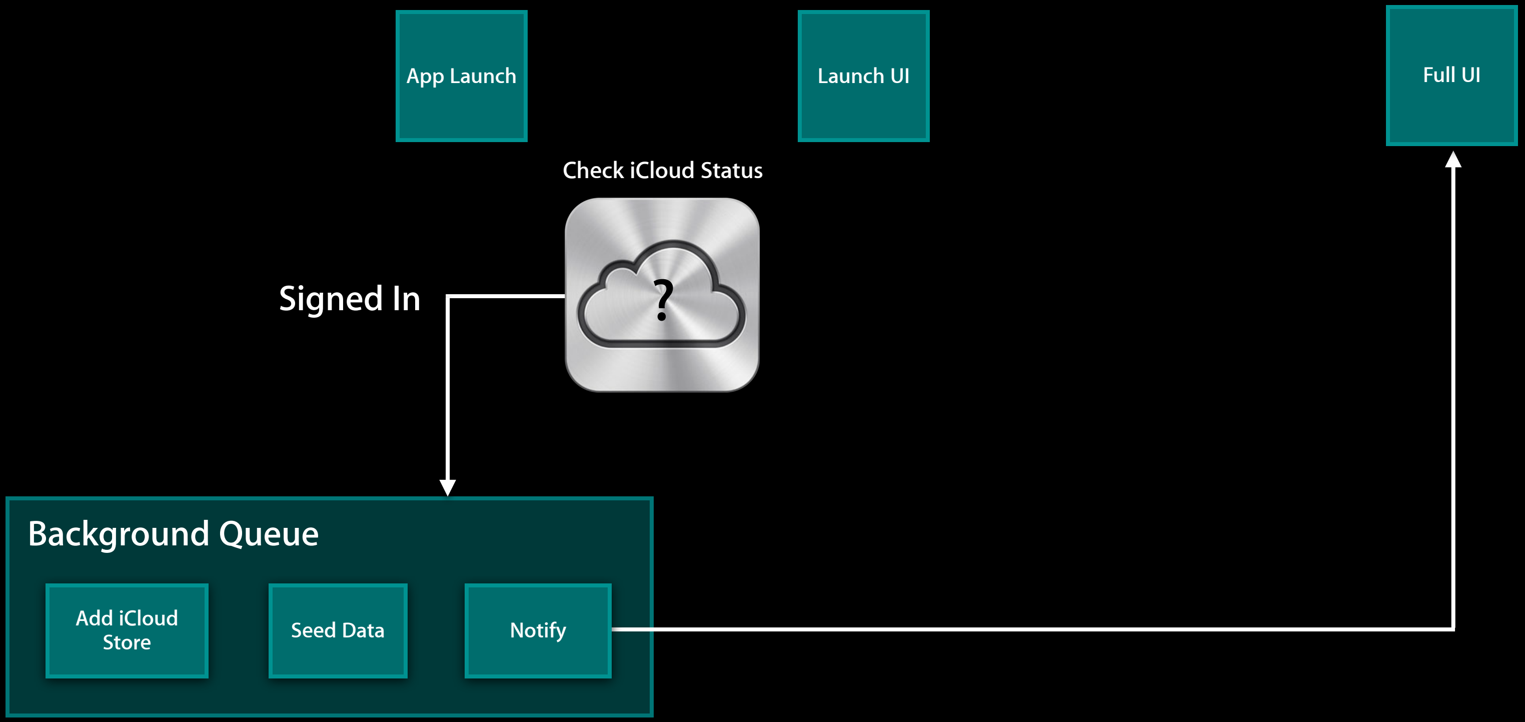
Background Queue

Add iCloud Store

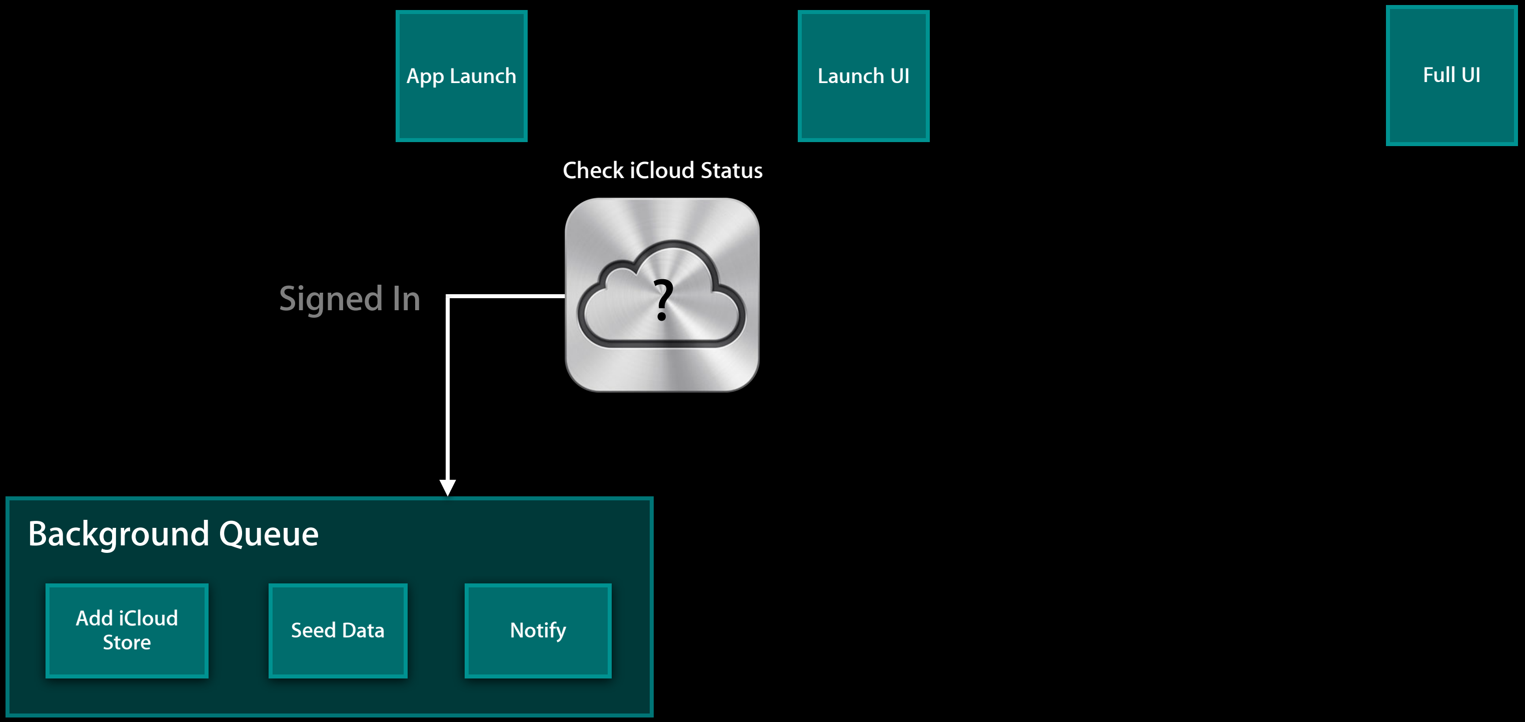
Seed Data

Notify

- Launch


App Launch
Launch UI
Full UI
Check iCloud Status
Signed In
Background Queue
Add iCloud Store
Seed Data
Notify

- Launch



App Launch

Launch UI

Full UI

Check iCloud Status

Signed In

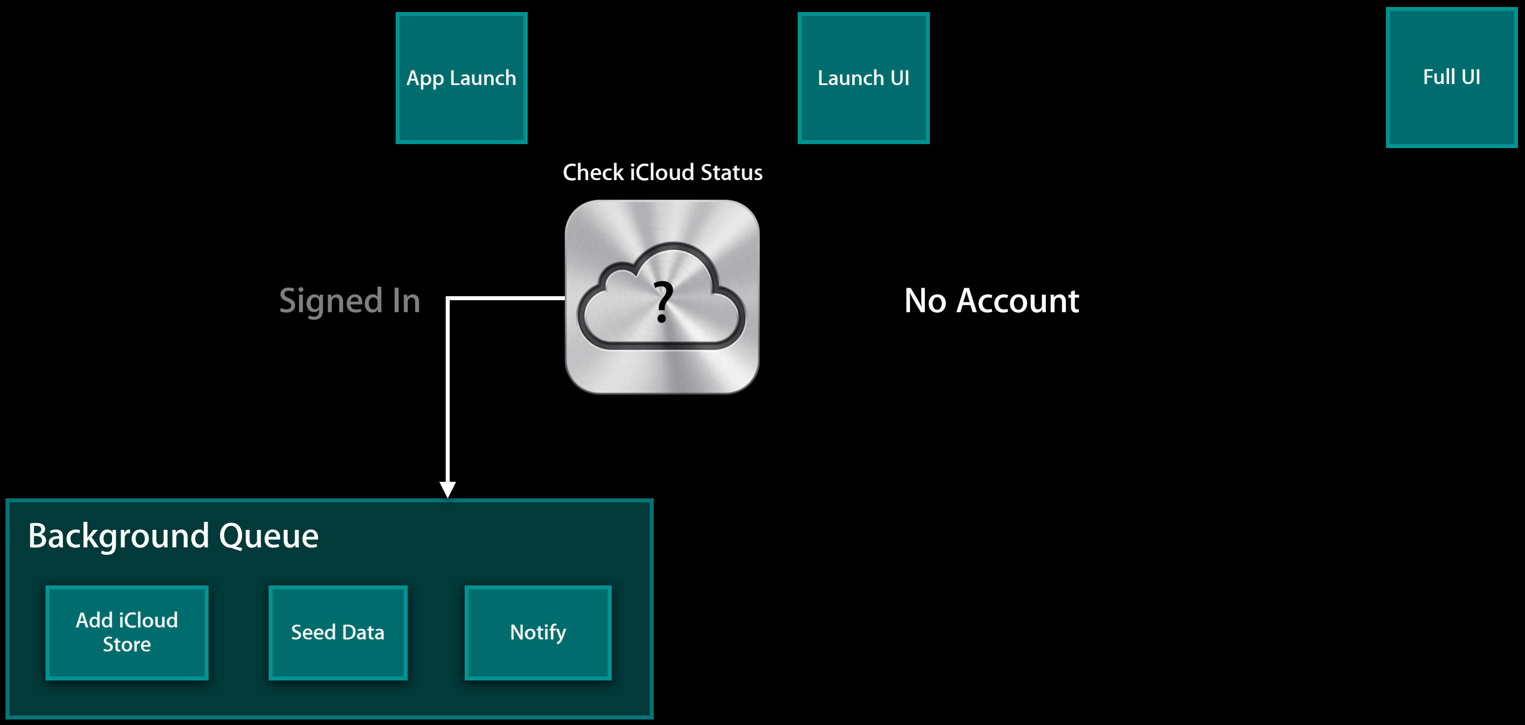
Background Queue

Add iCloud Store

Seed Data

Notify

- Launch



App Launch

Launch UI

Full UI

Check iCloud Status

?

Signed In

No Account

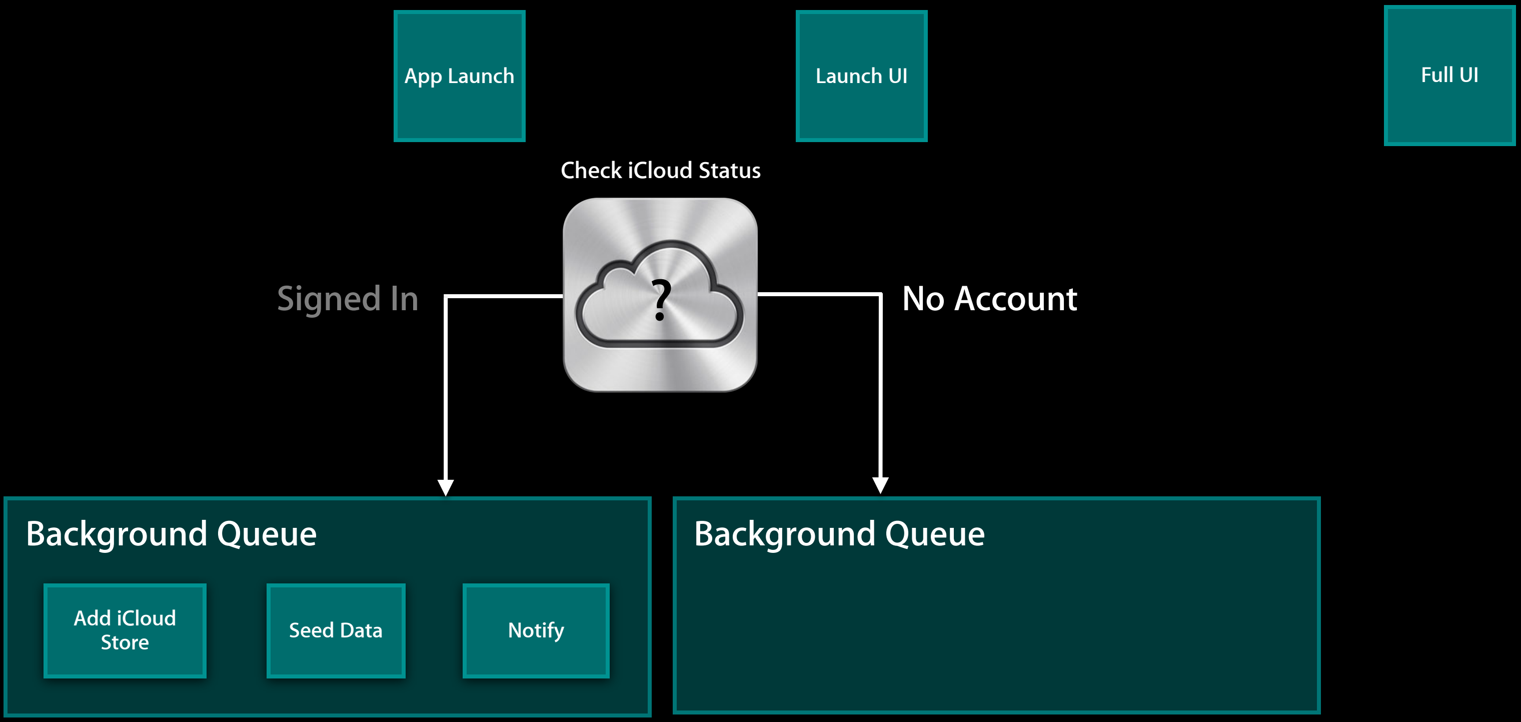
Background Queue

Add iCloud Store

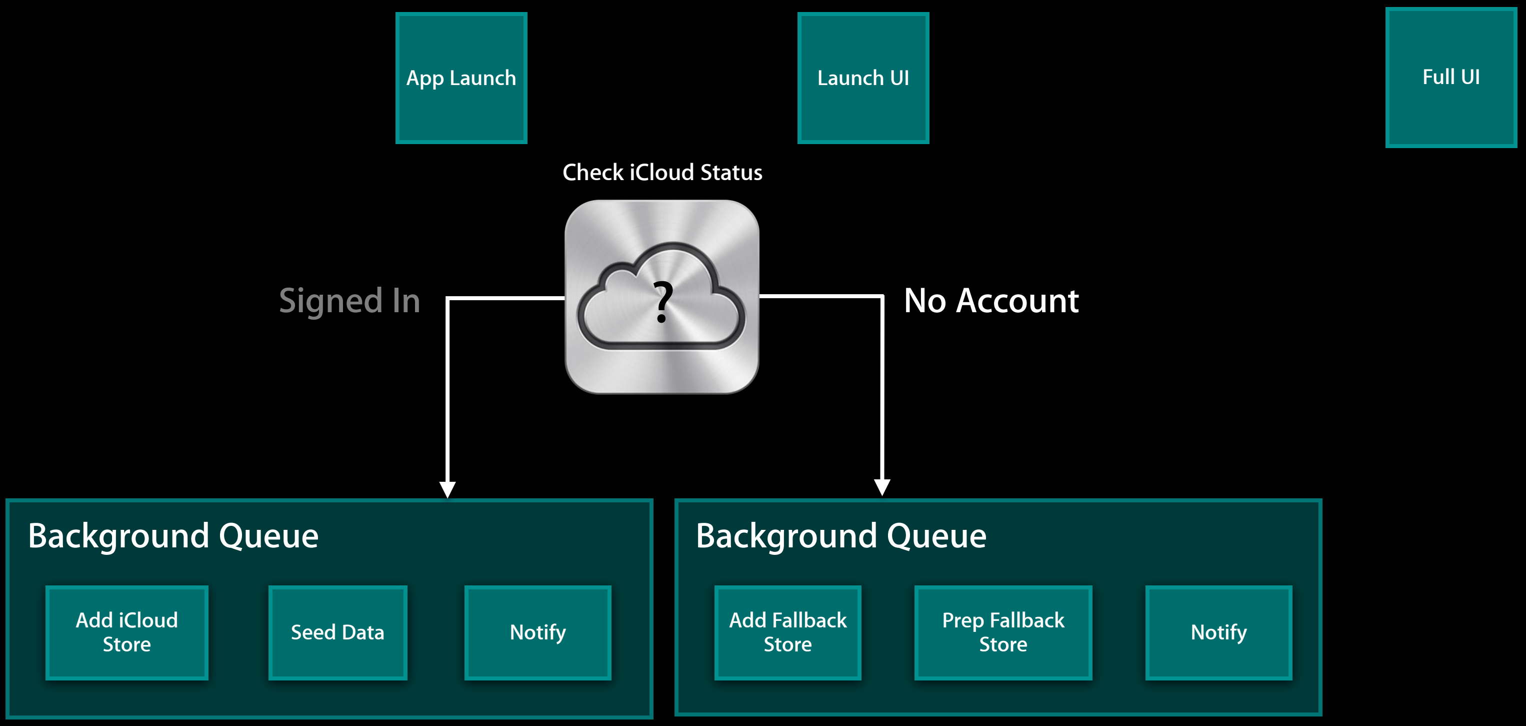
Seed Data

Notify

- Launch


App Launch
Launch UI
Full UI
Check iCloud Status
?
Signed In
No Account
Background Queue
Add iCloud Store
Seed Data
Notify
Background Queue

- Launch



App Launch

Launch UI

Full UI

Check iCloud Status

Signed In

No Account

Background Queue

Add iCloud Store

Seed Data

Notify

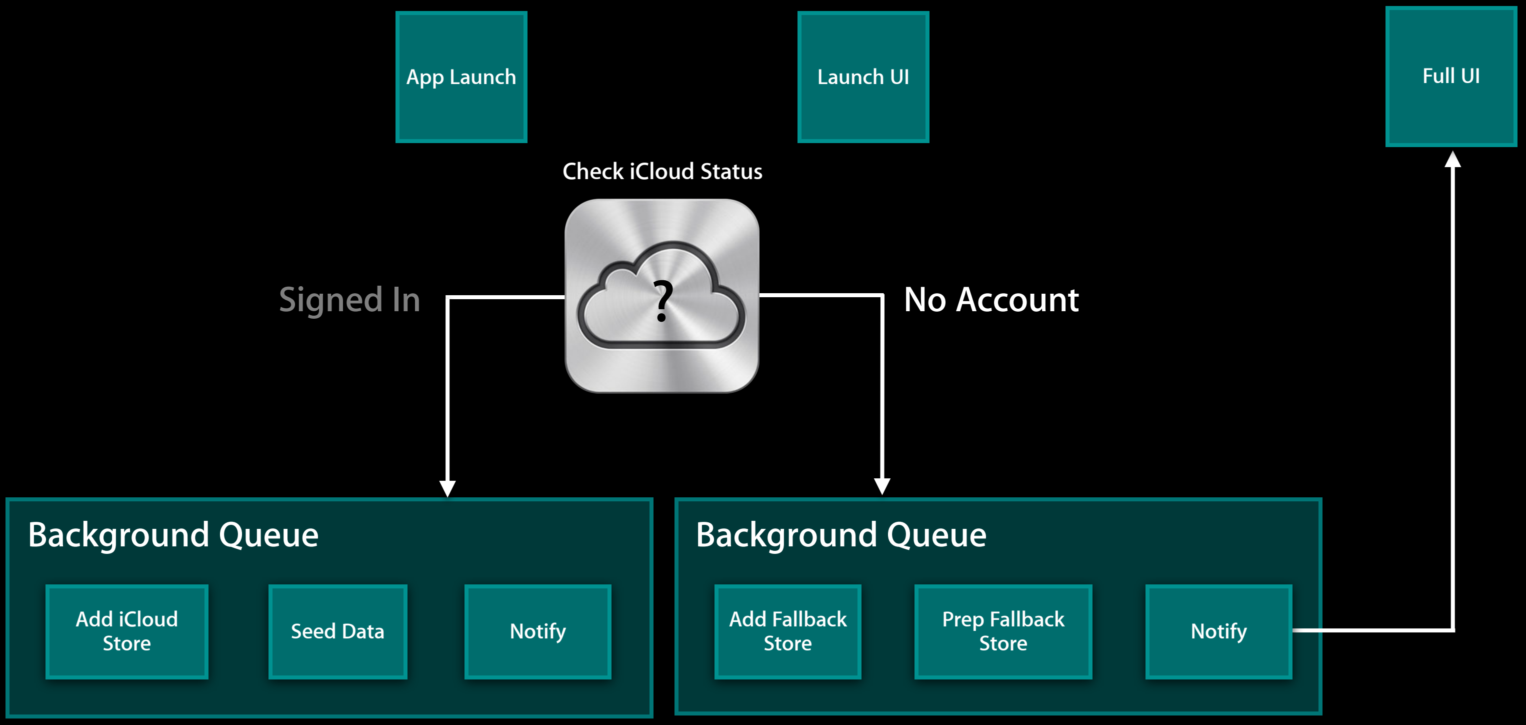
Background Queue

Add Fallback Store

Prep Fallback Store

Notify

- Launch


App Launch
Launch UI
Full UI
Check iCloud Status
?
Signed In
No Account
Background Queue
Add iCloud Store
Seed Data
Notify
Background Queue
Add Fallback Store
Prep Fallback Store
Notify

-init

- Launch
- Check iCloud Status

```
// Get token for iCloud user account
id currentToken = [fileManager ubiquityIdentityToken];
isiCloudSignedIn = (currentToken != nil);
```

–init

- Launch
- Check iCloud Status

```
// Get token for iCloud user account
id currentToken = [fileManager ubiquityIdentityToken];
isiCloudSignedIn = (currentToken != nil);
```

–loadPersistentStores

- Launch
- Check iCloud Status
- Add Persistent Store

```
- (void)loadPersistentStores {

    queue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
    dispatch_async(queue, ^(void) {
       [self asyncLoadPersistentStores];
    });
}
```

-loadiCloudStore

- Launch
- Check iCloud Status
- Add Persistent Store

```
- (BOOL)loadiCloudStore {
    NSFileManager *fm = [[NSFileManager alloc] init];
    _ubiquityURL = [fm URLForUbiquityContainerIdentifier:nil];

    iCloudStoreURL = [self iCloudStoreURL];
    iCloudDataURL = [ubiquityURL URLByAppendingPathComponent:@"iCloudData"];
    ...
    NSDictionary *options = @{
      NSPersistentStoreUbiquitousContentNameKey : @"iCloudStore"
      NSPersistentStoreUbiquitousContentURLKey : iCloudDataURL };
    [psc addPersistentStoreWithType: NSSQLiteStoreType
                      configuration: @”CloudConfig”
                                URL: iCloudStoreURL
                            options: options
                              error: &localError];
```

-loadiCloudStore

- Launch
- Check iCloud Status
- Add Persistent Store

```
- (BOOL)loadiCloudStore {
   NSFileManager *fm = [[NSFileManager alloc] init];
   _ubiquityURL = [fm URLForUbiquityContainerIdentifier:nil];

   iCloudStoreURL = [self iCloudStoreURL];
   iCloudDataURL = [ubiquityURL URLByAppendingPathComponent:@"iCloudData"];
   ...
   NSDictionary *options = @{
     NSPersistentStoreUbiquitousContentNameKey : @"iCloudStore"
     NSPersistentStoreUbiquitousContentURLKey : iCloudDataURL };
   [psc addPersistentStoreWithType: NSSQLiteStoreType
                     configuration: @”CloudConfig”
                               URL: iCloudStoreURL
                           options: options
                             error: &localError];
```

-loadiCloudStore

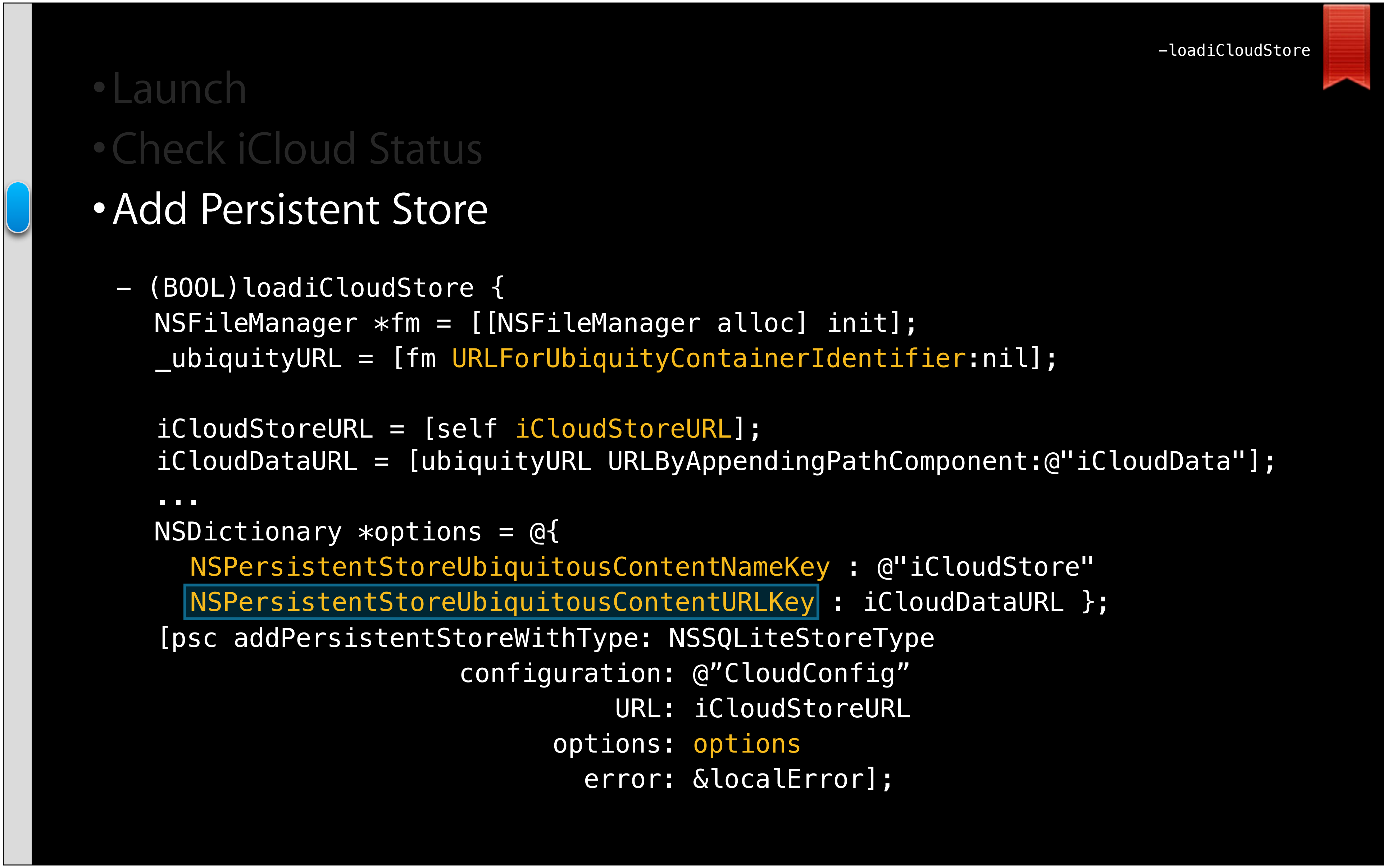
- Launch
- Check iCloud Status
- Add Persistent Store

```
- (BOOL)loadiCloudStore {
   NSFileManager *fm = [[NSFileManager alloc] init];
   _ubiquityURL = [fm URLForUbiquityContainerIdentifier:nil];

   iCloudStoreURL = [self iCloudStoreURL];
   iCloudDataURL = [ubiquityURL URLByAppendingPathComponent:@"iCloudData"];
   ...
   NSDictionary *options = @{
     NSPersistentStoreUbiquitousContentNameKey : @"iCloudStore"
     NSPersistentStoreUbiquitousContentURLKey : iCloudDataURL };
   [psc addPersistentStoreWithType: NSSQLiteStoreType
                     configuration: @”CloudConfig”
                               URL: iCloudStoreURL
                           options: options
                             error: &localError];
```

-loadiCloudStore

- Launch
- Check iCloud Status
- Add Persistent Store

```
- (BOOL)loadiCloudStore {
   NSFileManager *fm = [[NSFileManager alloc] init];
   _ubiquityURL = [fm URLForUbiquityContainerIdentifier:nil];

   iCloudStoreURL = [self iCloudStoreURL];
   iCloudDataURL = [ubiquityURL URLByAppendingPathComponent:@"iCloudData"];
   ...
   NSDictionary *options = @{
     NSPersistentStoreUbiquitousContentNameKey : @"iCloudStore"
     NSPersistentStoreUbiquitousContentURLKey : iCloudDataURL };
   [psc addPersistentStoreWithType: NSSQLiteStoreType
                     configuration: @”CloudConfig”
                               URL: iCloudStoreURL
                           options: options
                             error: &localError];
```



-loadiCloudStore

- Launch
- Check iCloud Status
- Add Persistent Store

```
- (BOOL)loadiCloudStore {
   NSFileManager *fm = [[NSFileManager alloc] init];
   _ubiquityURL = [fm URLForUbiquityContainerIdentifier:nil];

   iCloudStoreURL = [self iCloudStoreURL];
   iCloudDataURL = [ubiquityURL URLByAppendingPathComponent:@"iCloudData"];
   ...
   NSDictionary *options = @{
     NSPersistentStoreUbiquitousContentNameKey : @"iCloudStore"
     NSPersistentStoreUbiquitousContentURLKey : iCloudDataURL };
   [psc addPersistentStoreWithType: NSSQLiteStoreType
                      configuration: @”CloudConfig”
                                URL: iCloudStoreURL
                            options: options
                              error: &localError];
```

-loadiCloudStore

- Launch
- Check iCloud Status
- Add Persistent Store

```
- (BOOL)loadiCloudStore {
  NSFileManager *fm = [[NSFileManager alloc] init];
  _ubiquityURL = [fm URLForUbiquityContainerIdentifier:nil];

  iCloudStoreURL = [self iCloudStoreURL];
  iCloudDataURL = [ubiquityURL URLByAppendingPathComponent:@"iCloudData"];
  ...
  NSDictionary *options = @{
    NSPersistentStoreUbiquitousContentNameKey : @"iCloudStore"
    NSPersistentStoreUbiquitousContentURLKey : iCloudDataURL };
  [psc addPersistentStoreWithType: NSSQLiteStoreType
                      configuration: @”CloudConfig”
                                URL: iCloudStoreURL
                            options: options
                              error: &localError];
```

-asyncLoadPersistenStores

- Launch
- Check iCloud Status
- Add Persistent Store
- Seed Initial Data

```
- (void)asyncLoadPersistentStores {
    ...

    if ([self loadiCloudStore]) {
       ...
       [self seedPersistentStore:_iCloudStore
        withPersistentStoreAtURL:[self seedStoreURL]
                                error:&error];
    }
}
```

-application:didFinishLaunchingWithOptions:

- Launch
- Check iCloud Status
- Add Persistent Store
- Seed Initial Data
- Notify Store Ready

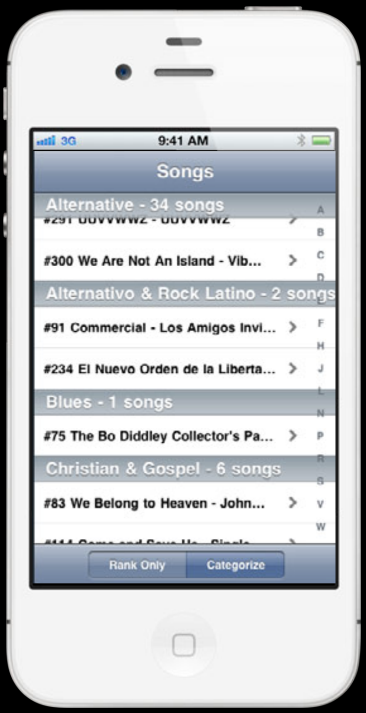
```
[[NSNotificationCenter defaultCenter]
          addObserver: rootViewController
             selector: @selector(reloadFetchedResults:)
                 name: NSPersistentStoreCoordinatorStoresDidChangeNotification
               object: psc];
```

-application:didFinishLaunchingWithOptions:

- Launch
- Check iCloud Status
- Add Persistent Store
- Seed Initial Data
- Notify Store Ready

```
[[NSNotificationCenter defaultCenter]
           addObserver: rootViewController
              selector: @selector(reloadFetchedResults:)
                  name: NSPersistentStoreCoordinatorStoresDidChangeNotification
                object: psc];
```

- Launch
- Check iCloud Status
- Add Persistent Store
- Seed Initial Data
- Notify Store Ready
- Full UI!



Songs

Alternative - 34 songs

#291 UUVVWWZ - UUVVWWZ

#300 We Are Not An Island - Vib...

Alternativo & Rock Latino - 2 songs

#91 Commercial - Los Amigos Invi...

#234 El Nuevo Orden de la Liberta...

Blues - 1 songs

#75 The Bo Diddley Collector's Pa...

Christian & Gospel - 6 songs

#83 We Belong to Heaven - John...

#114 Come and Save Us - Single

Design

Launch

Lifecycle

Design

Launch

Lifecycle

Application Lifecycle + iCloud

Application Lifecycle + iCloud

- Seeding
- Integrate changes
- Responding to User Events
- Performance
- Debugging

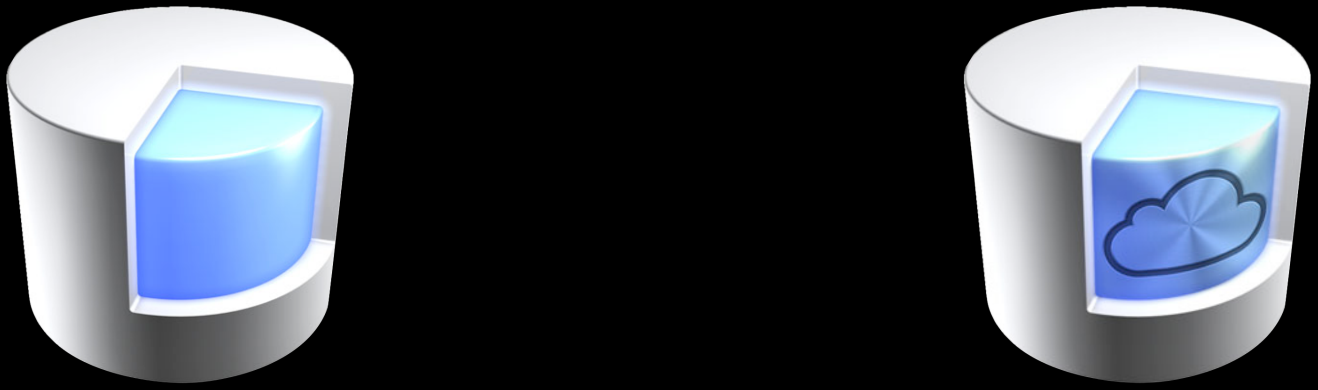
Demo

Sample application

Seeding

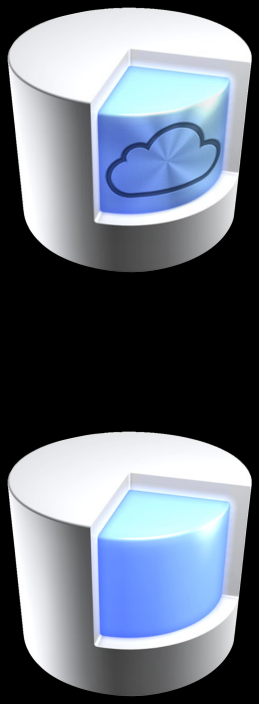
Seeding

Seeding



Seeding

- Add Seed Store



Seeding

- Add Seed Store

```
NSPersistentStoreUbiquitousContentNameKey
NSPersistentStoreUbiquitousContentURLKey
```

Seeding

- Add Seed Store

```
NSPersistentStoreUbiquitousContentNameKey
NSPersistentStoreUbiquitousContentURLKey
```

```
-addPersistentStore:
```

```
NSReadOnlyPersistentStoreOption
```

-seedPersistentStoreStore:
withPersistentStoreAtURL:
error:

Seeding

- Add Seed Store
- Migrate objects

```
NSUInteger batchSize = 500;
[fr setFetchBatchSize:batchSize];

seedObjs = [moc executeFetchRequest:fr error:&error];
```

-seedPersistentStoreStore:
withPersistentStoreAtURL:
error:

Seeding

- Add Seed Store
- Migrate objects

```
NSUInteger batchSize = 500;
[fr setFetchBatchSize:batchSize];

seedObjs = [moc executeFetchRequest:fr error:&error];

for (NSManagedObject *obj in seedObjs) {
    [self addManagedObjectToiCloudStore:obj];

    if (0 == (i % batchSize)) {
        if ([moc save:&error]) {
            [moc reset];
        }
    }

    i++;
}
```

Seeding

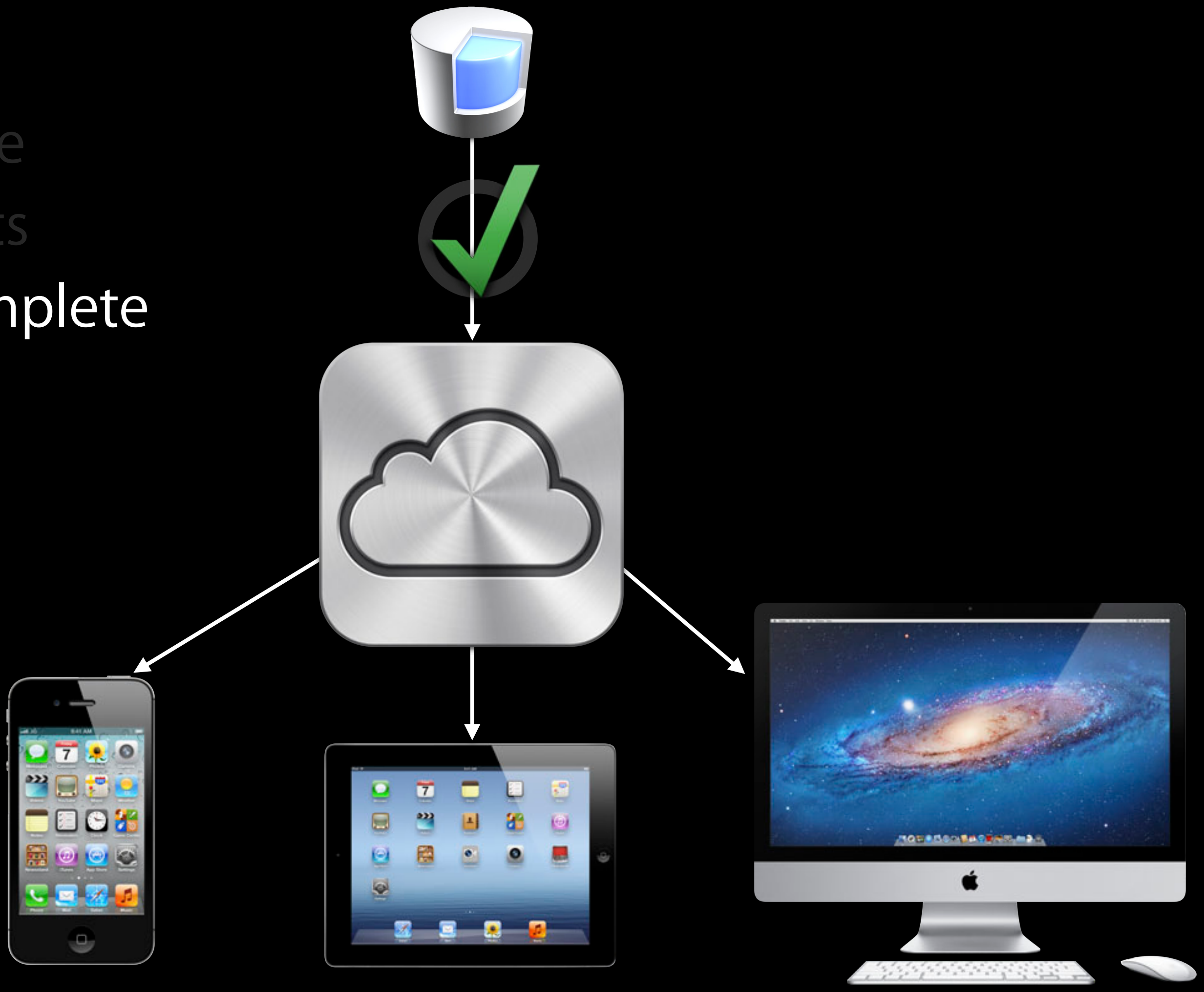
- Add Seed Store
- Migrate objects
- Mark seed complete

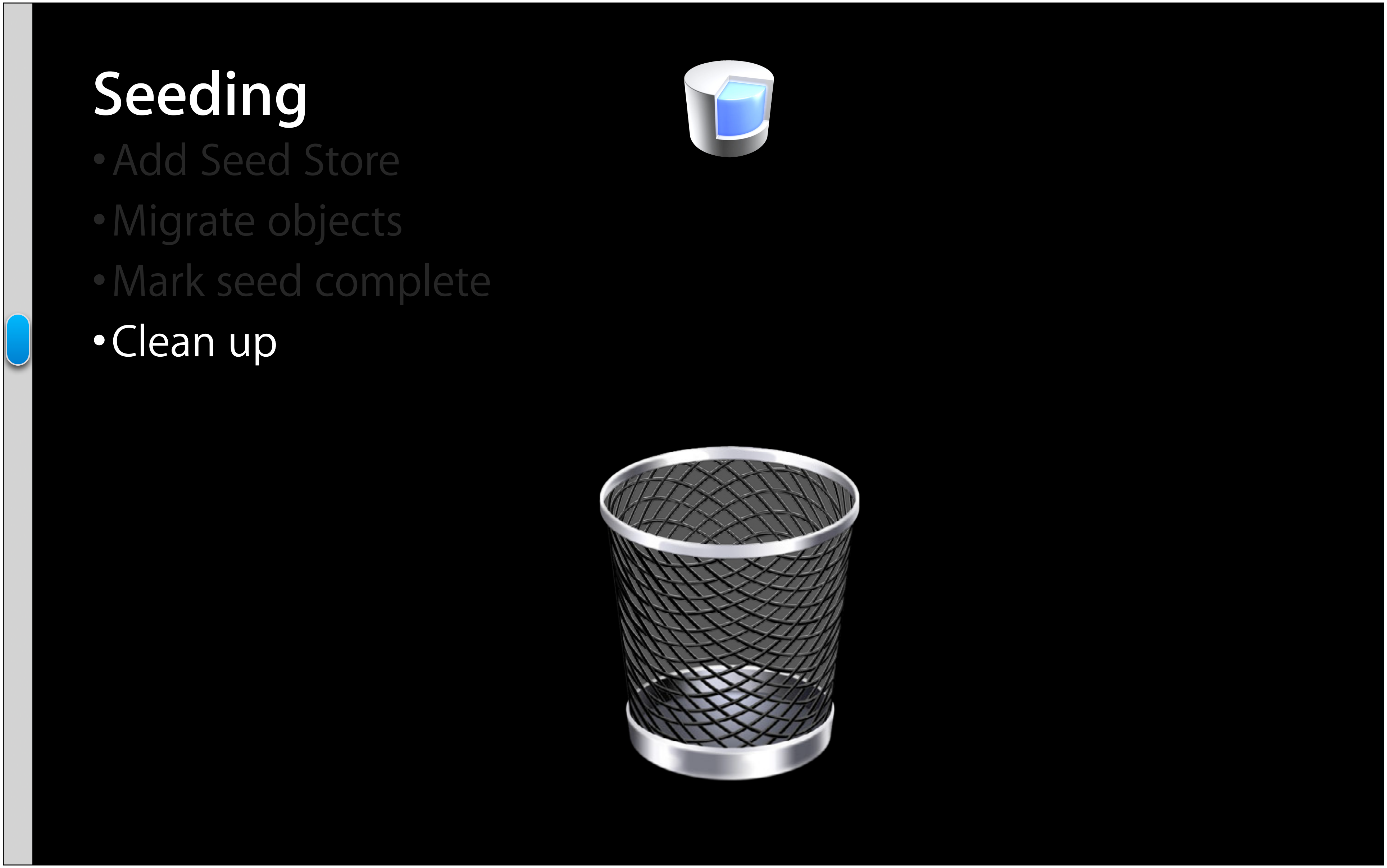
Seeding

- Add Seed Store
- Migrate objects
- Mark seed complete

Seeding

- Add Seed Store
- Migrate objects
- Mark seed complete





Seeding

- Add Seed Store
- Migrate objects
- Mark seed complete
- Clean up

Seeding

- Add Seed Store
- Migrate objects
- Mark seed complete
- Clean up

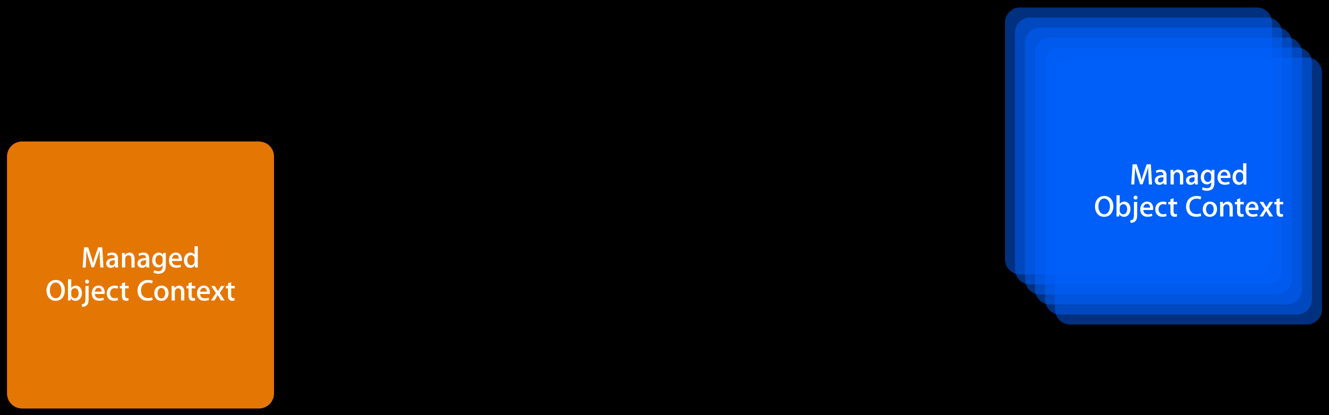
Demo

Seeding

Lifecycle

Integrating Changes

Change Integration

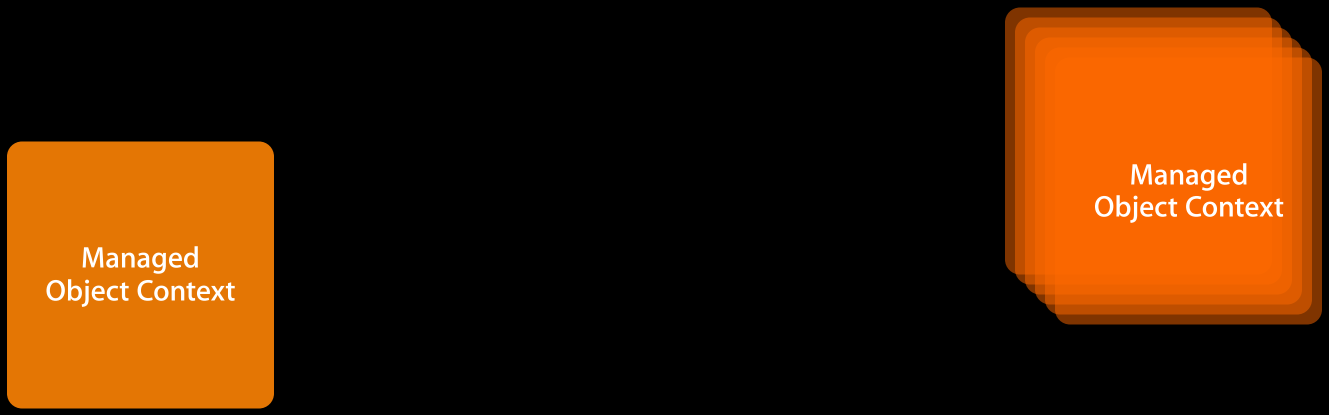


Managed Object Context

Managed Object Context

Change Integration

`NSManagedObjectContextDidSaveNotification`

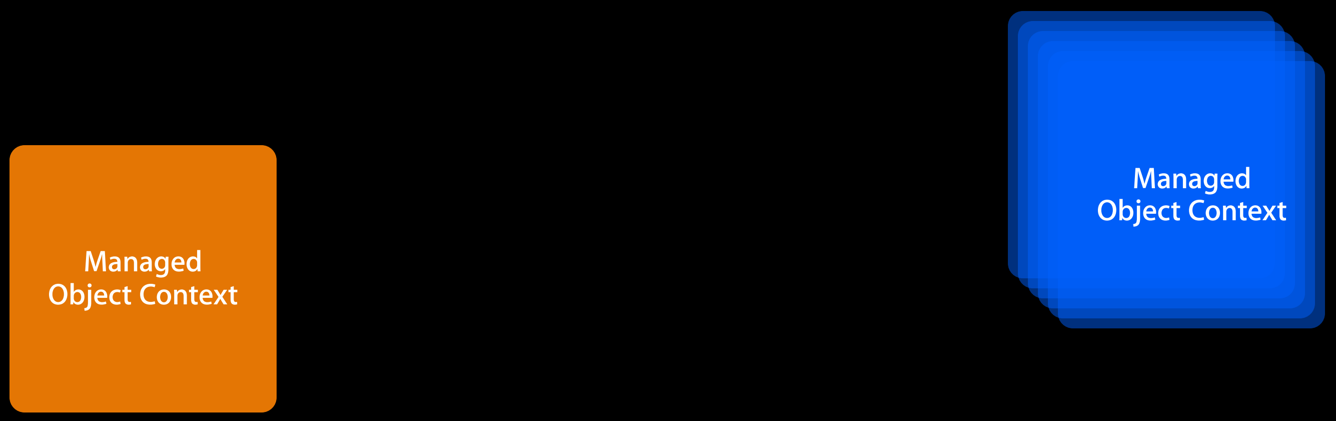


Managed Object Context

Managed Object Context

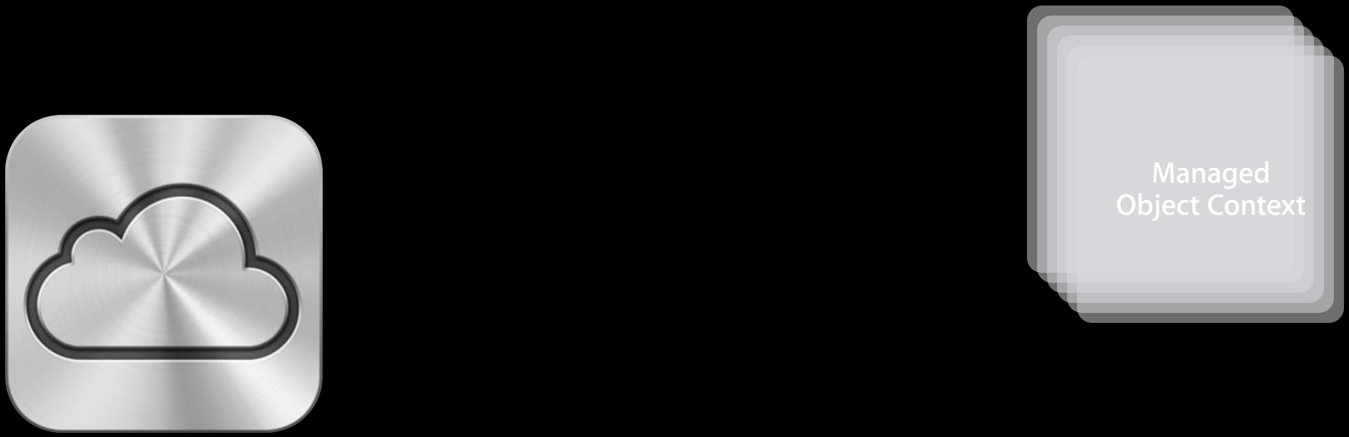
Change Integration

`NSManagedObjectContextDidSaveNotification`



Change Integration

```
NSManagedObjectContextDidSaveNotification
NSPersistentStoreDidImportUbiquitousContentChangesNotification
```



Change Integration

Change Integration

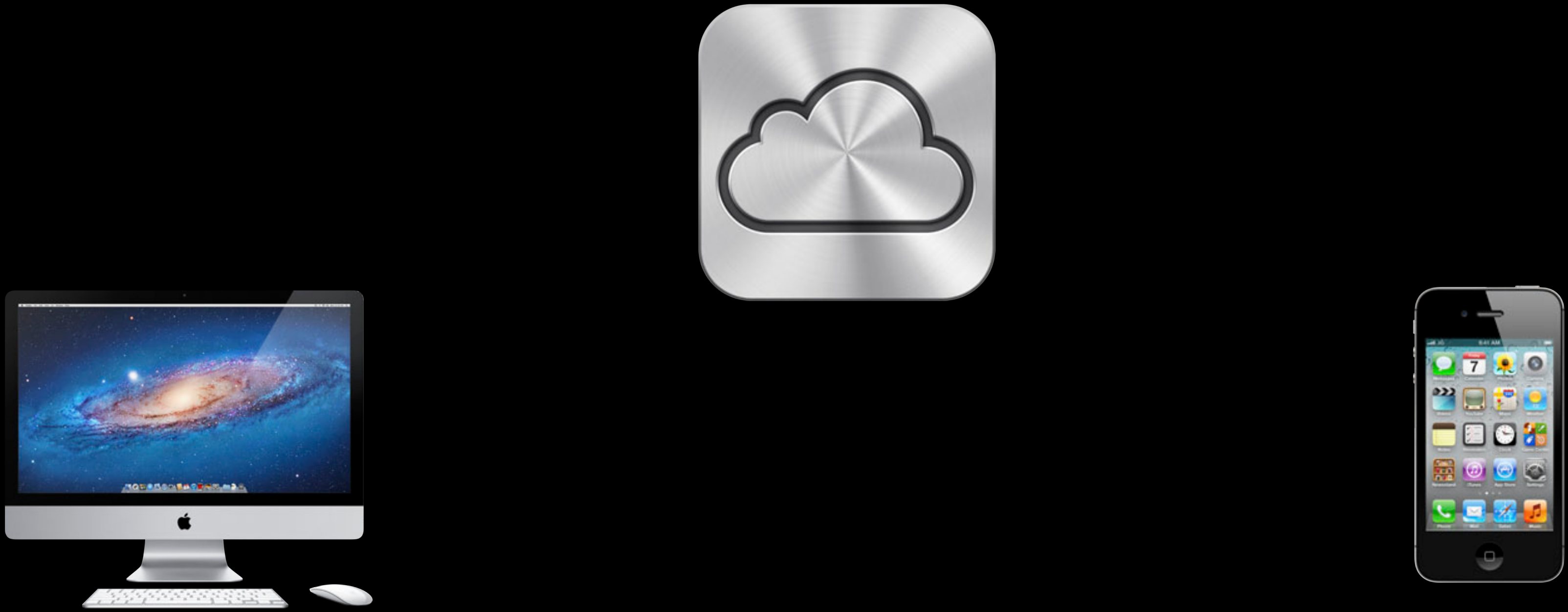
- `NSManagedObjectContextDidSaveNotification`
    - `NSManagedObjects`

Change Integration

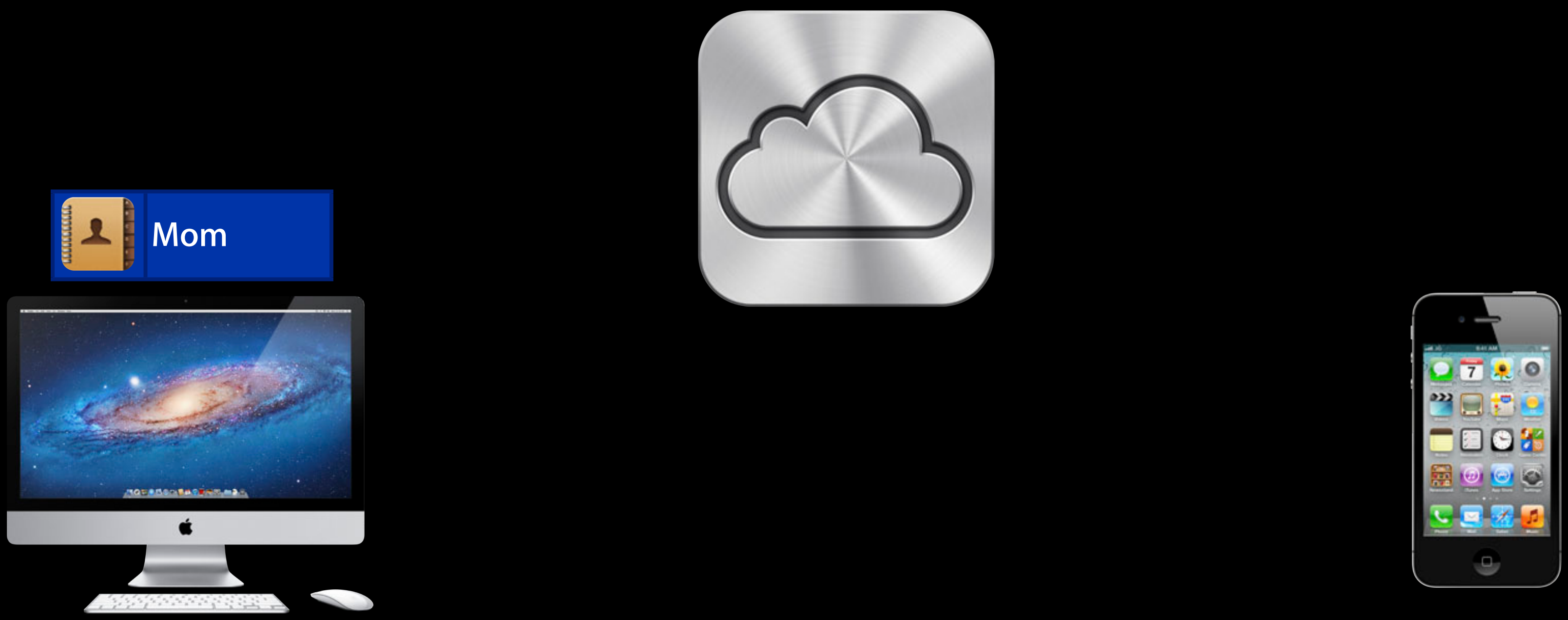
- NSManagedObjectContextDidSaveNotification
    - NSManagedObjects
- NSPersistentStoreDidImportUbiquitousContentChangesNotification
    - NSManagedObjectIDs

Integrating Changes

Integrating Changes

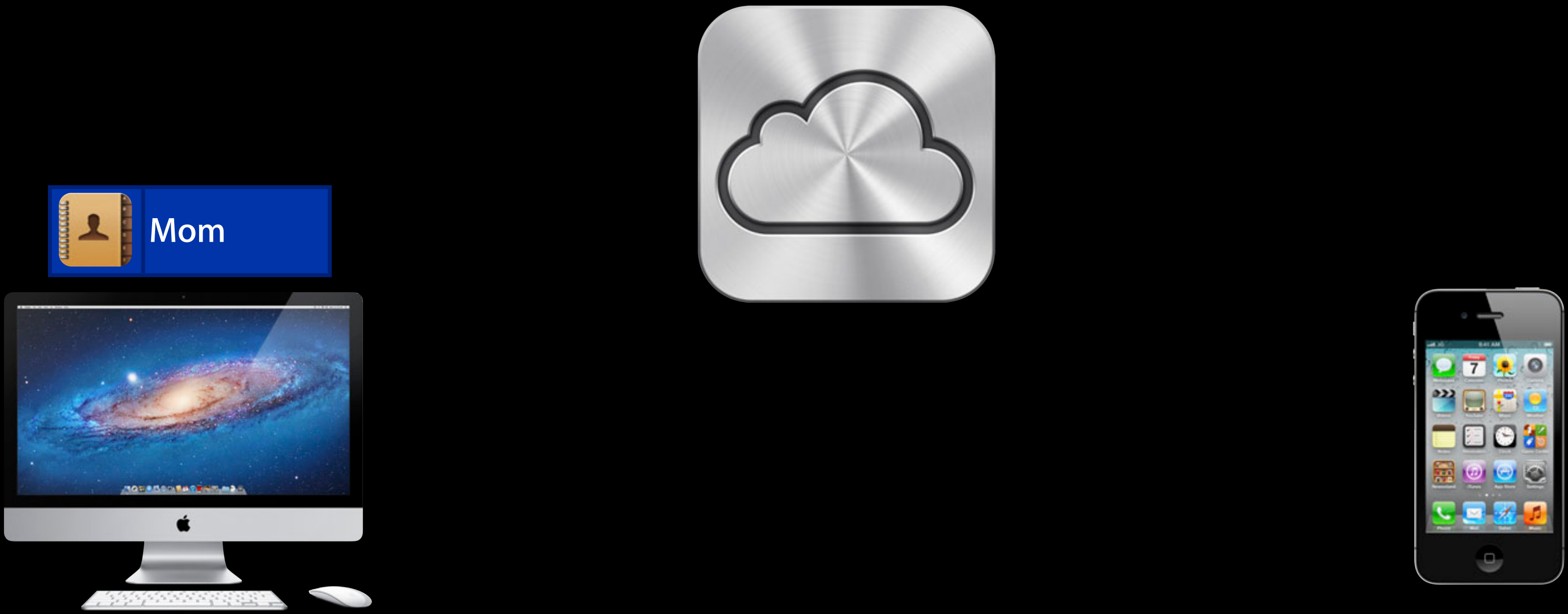


Integrating Changes

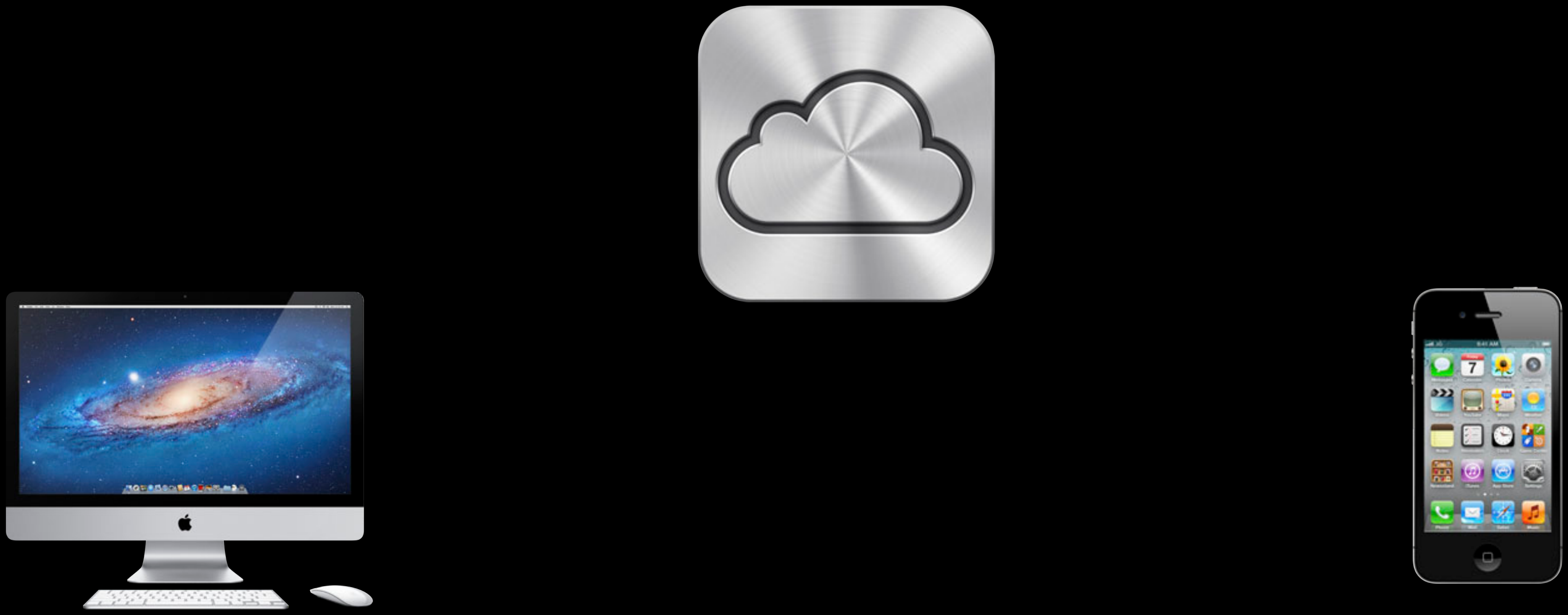


Mom

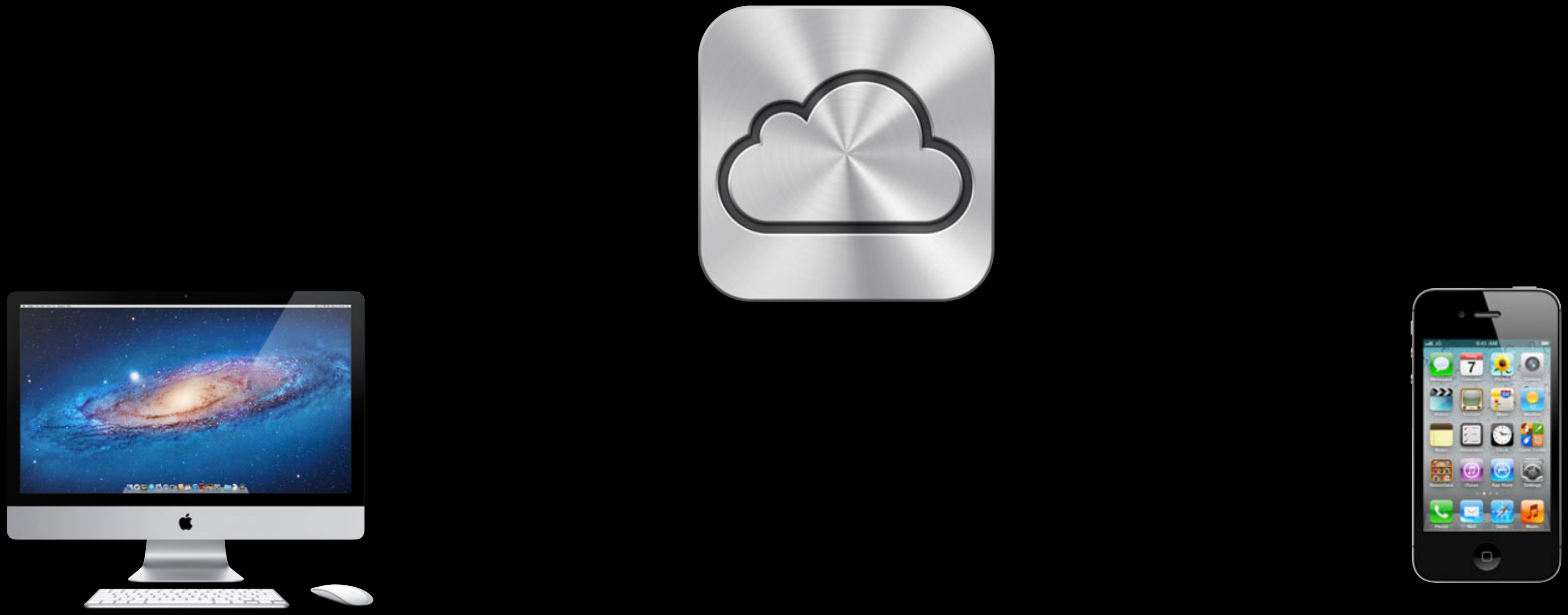
Integrating Changes



Integrating Changes



Integrating Changes



Integrating Changes

PK	Contact
1	Mom
2	Mom

Uniquing

- Find the Duplicates

-deDupe:

Uniquing

- Find the Duplicates

```
select zemailaddress, count(zemailaddress)
from zperson group by zemailaddress;
```

Uniquing

- Find the Duplicates

```
select zemailaddress, count(zemailaddress)
from zperson group by zemailaddress;
```

zemailddress	count(zemailaddress)
cd@wwdc.com	1
core@data.com	1
moc@save.com	256

Uniquing

- Find the Duplicates

Uniquing

- Find the Duplicates

```
NSExpression *countExpr = [NSExpression expressionWithFormat:@"count:(emailAddress)"];
```

-deDupe:

Uniquing

- Find the Duplicates

```
NSExpression *countExpr = [NSExpression expressionWithFormat:@"count:(emailAddress)"];

NSAttributeDescription *emailAttr;
NSFetchRequest *fr;
[fr setPropertiesToFetch:[NSArray arrayWithObjects:emailAttr, countExpr, nil]];
[fr setPropertiesToGroupBy:[NSArray arrayWithObject:emailAttr]];
```

-deDupe:

Uniquing

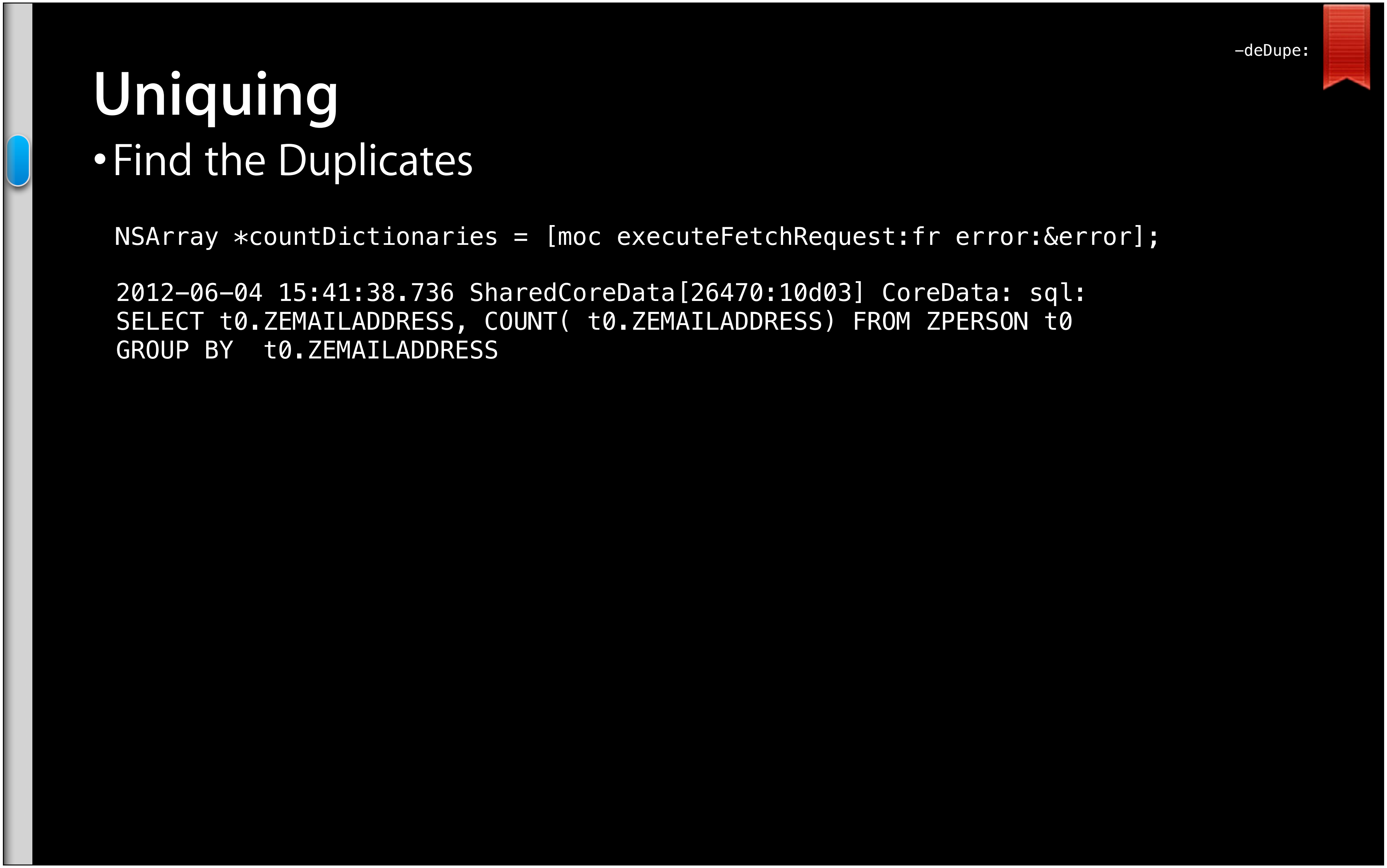
- Find the Duplicates

```
NSExpression *countExpr = [NSExpression expressionWithFormat:@"count:(emailAddress)"];

NSAttributeDescription *emailAttr;
NSFetchRequest *fr;
[fr setPropertiesToFetch:[NSArray arrayWithObjects:emailAttr, countExpr, nil]];
[fr setPropertiesToGroupBy:[NSArray arrayWithObject:emailAttr]];

[fr setResultType:NSDictionaryResultType];
```

-deDupe:

Uniquing

- Find the Duplicates



-deDupe:

Uniquing

- Find the Duplicates

```
NSArray *countDictionaries = [moc executeFetchRequest:fr error:&error];

2012-06-04 15:41:38.736 SharedCoreData[26470:10d03] CoreData: sql:
SELECT t0.ZEMAILADDRESS, COUNT( t0.ZEMAILADDRESS) FROM ZPERSON t0
GROUP BY  t0.ZEMAILADDRESS
```

Uniquing

- Find the Duplicates

```
NSArray *countDictionaries = [moc executeFetchRequest:fr error:&error];

2012-06-04 15:41:38.736 SharedCoreData[26470:10d03] CoreData: sql:
SELECT t0.ZEMAILADDRESS, COUNT( t0.ZEMAILADDRESS) FROM ZPERSON t0
GROUP BY  t0.ZEMAILADDRESS

(lldb) po countDictionaries
(NSArray *) $2 = 0x07c0cb80 <_PFArray 0x7c0cb80>(
{
    count = 1;
    emailAddress = "cd@wwdc.com";
},
{
    count = 256;
    emailAddress = "moc@save.com";
},
{
    count = 1;
    emailAddress = "core@data.com";
})
```

-deDupe:

Uniquing

- Find the Duplicates
- Fetch Duplicate Objects

-deDupe:

Uniquing

- Find the Duplicates
- Fetch Duplicate Objects

```
p = [NSPredicate predicateWithFormat:@"emailAddress IN (%@)", emailsWithDupes];
[fr setPredicate:p];
```

-deDupe:

Uniquing

- Find the Duplicates
- Fetch Duplicate Objects

```
p = [NSPredicate predicateWithFormat:@"emailAddress IN (%@)", emailsWithDupes];
[fr setPredicate:p];

emailSort = [NSSortDescriptor sortDescriptorWithKey:@"emailAddress"
                                          ascending:YES];
[fr setSortDescriptors:[NSArray arrayWithObject:emailSort]];

NSArray *dupes = [moc executeFetchRequest:fr error:&error];
```

-deDupe:

Uniquing

- Find the Duplicates
- Fetch Duplicate Objects
- Choose a winner

`-deDupe:`

Uniquing

- Find the Duplicates
- Fetch Duplicate Objects
- Choose a winner
    - Ensure consistent merges across peers

Uniquing

- Find the Duplicates
- Fetch Duplicate Objects
- Choose a winner
  - Ensure consistent merges across peers
  - Use a record UUID or timestamp

-deDupe:

Uniquing

- Find the Duplicates
- Fetch Duplicate Objects
- Choose a winner
  - Ensure consistent merges across peers
  - Use a record UUID or timestamp

```
for (NSManagedObject *dupe in duplicates) {
    //choose winner
    if (0 == (i % batchSize)) {
        [moc save:&error];
    }
    i++;
}
```

Demo

Uniquing

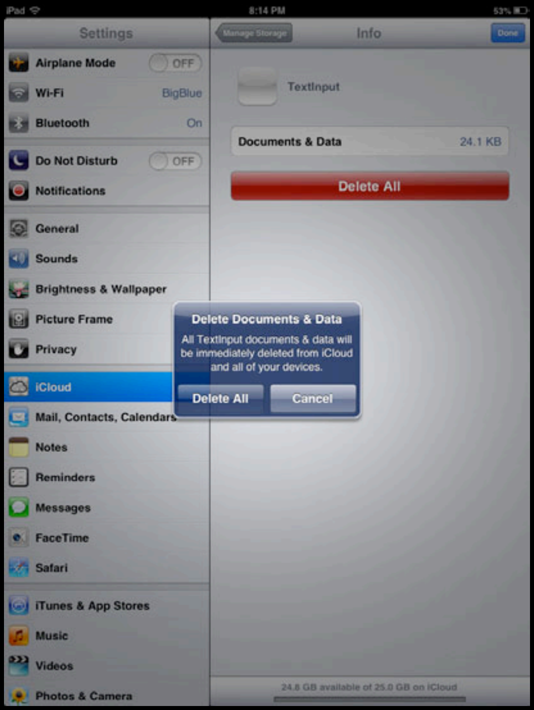
Lifecycle

User Events

User Events

User Events

- Delete from Documents & Data



Airplane Mode OFF

Wi-Fi BigBlue

Bluetooth On

Do Not Disturb OFF

TextInput

Documents & Data 24.1 KB

Delete Documents & Data

All TextInput documents & data will be immediately deleted from iCloud and all of your devices.

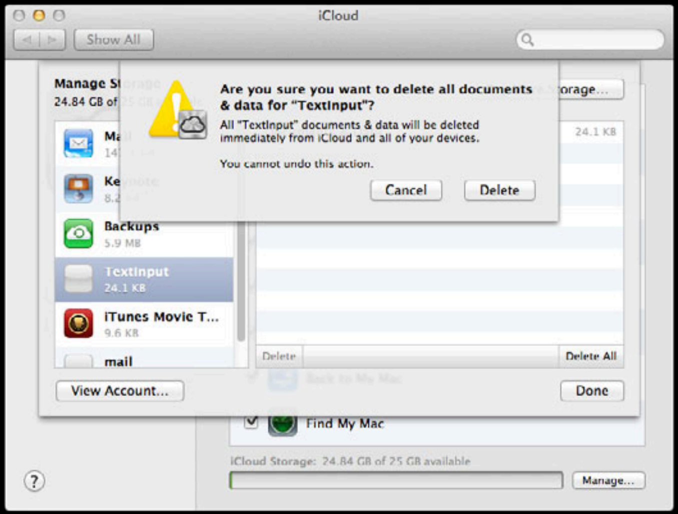
24.8 GB available of 25.0 GB on iCloud

User Events

- Delete from Documents & Data

User Events

- Delete from Documents & Data



Are you sure you want to delete all documents & data for "TextInput"?

All "TextInput" documents & data will be deleted immediately from iCloud and all of your devices.

You cannot undo this action.

User Events

- Delete from Documents & Data

User Events

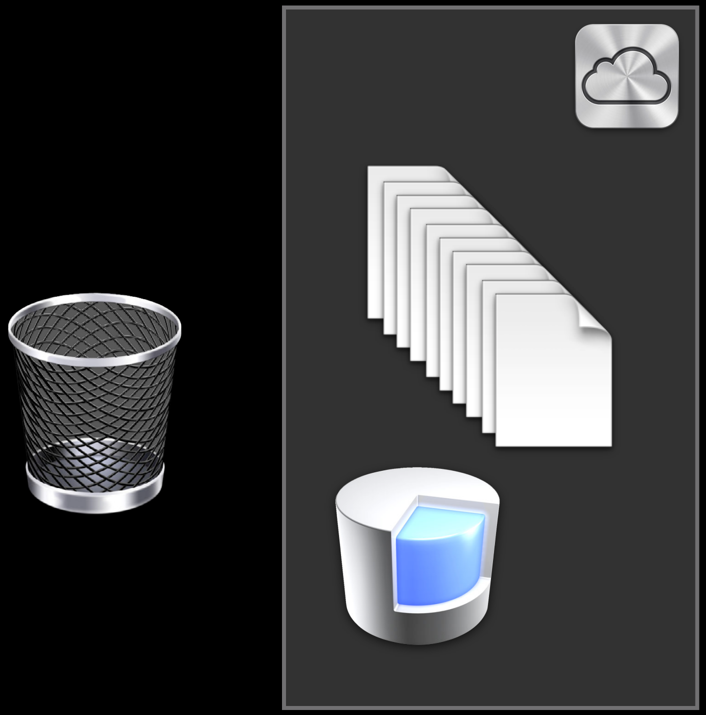
- Delete from Documents & Data

User Events

- Delete from Documents & Data

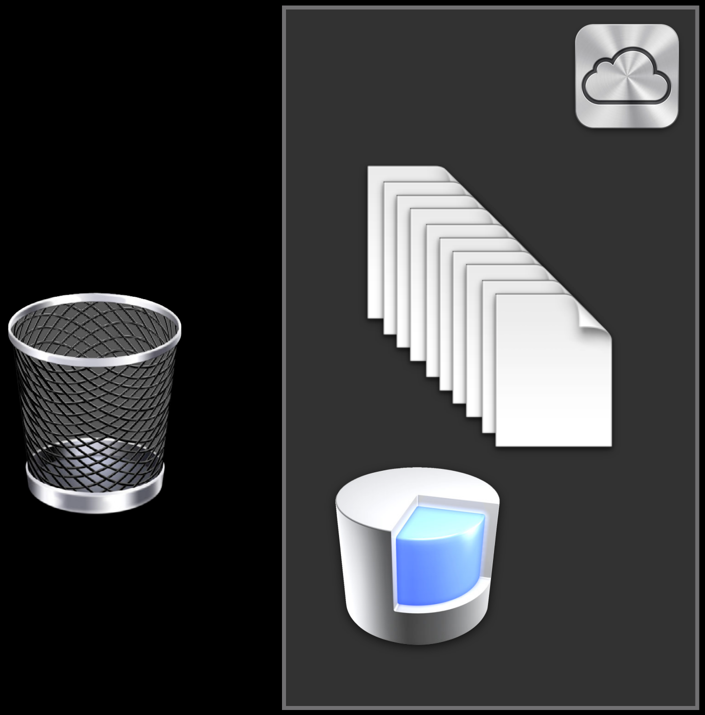
User Events

- Delete from Documents & Data



User Events

- Delete from Documents & Data



User Events

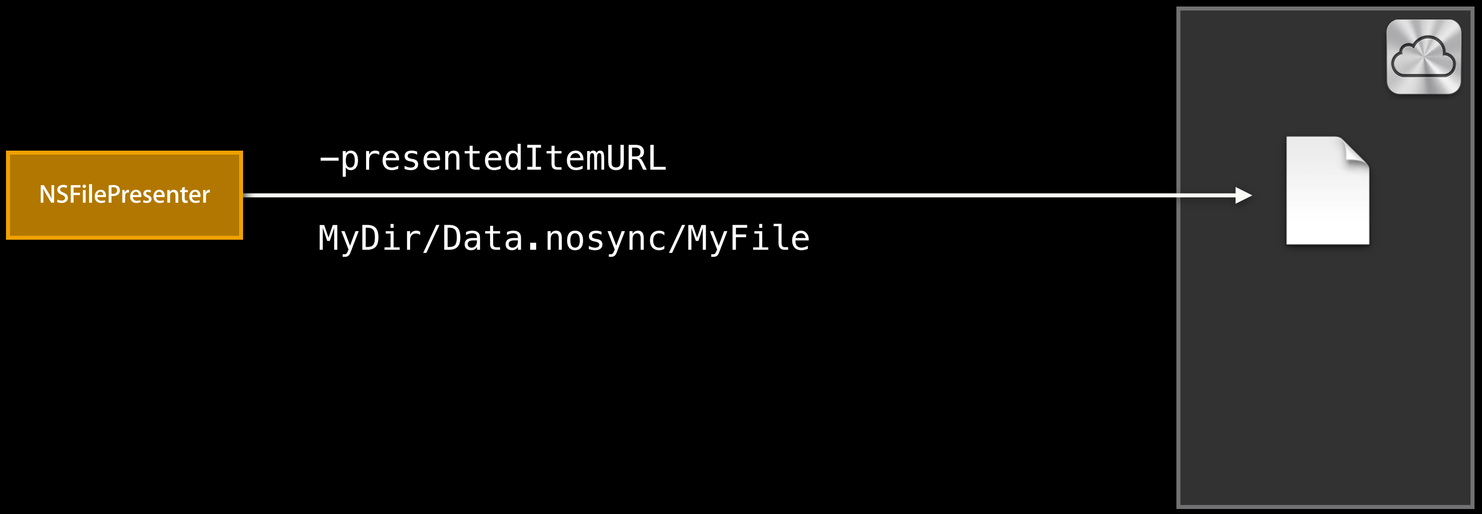
- Delete from Documents & Data

User Events

Documents & Data

User Events

Documents & Data



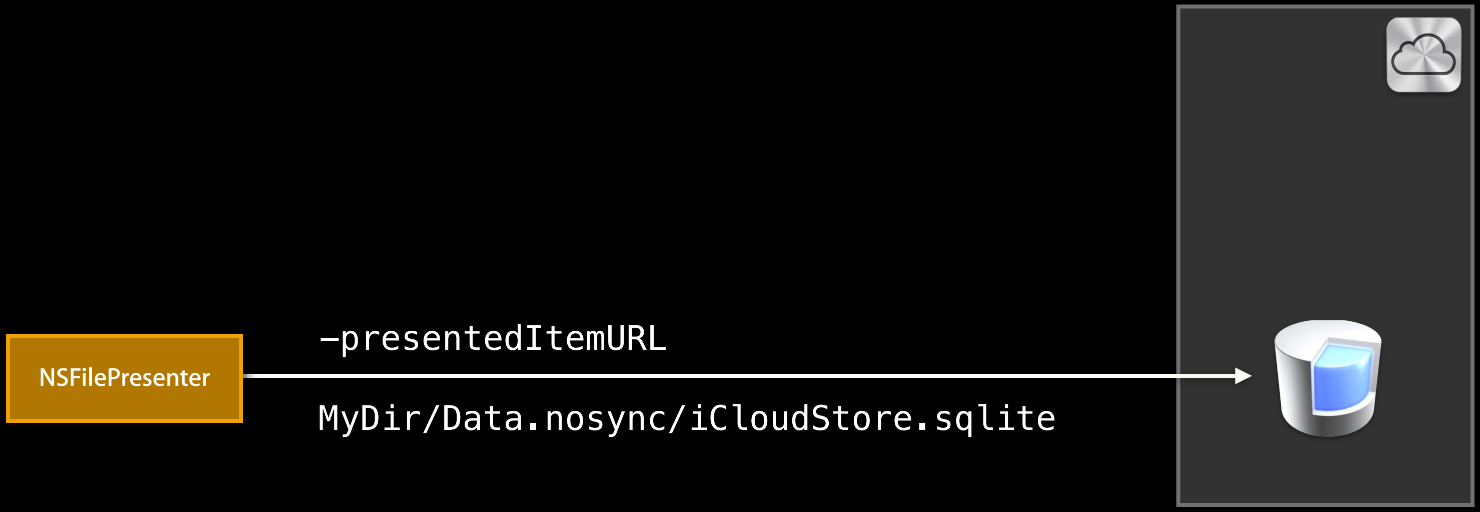
NSFilePresenter

`-presentedItemURL`

`MyDir/Data.nosync/MyFile`

User Events

Documents & Data



NSFilePresenter

`-presentedItemURL`

`MyDir/Data.nosync/iCloudStore.sqlite`

User Events

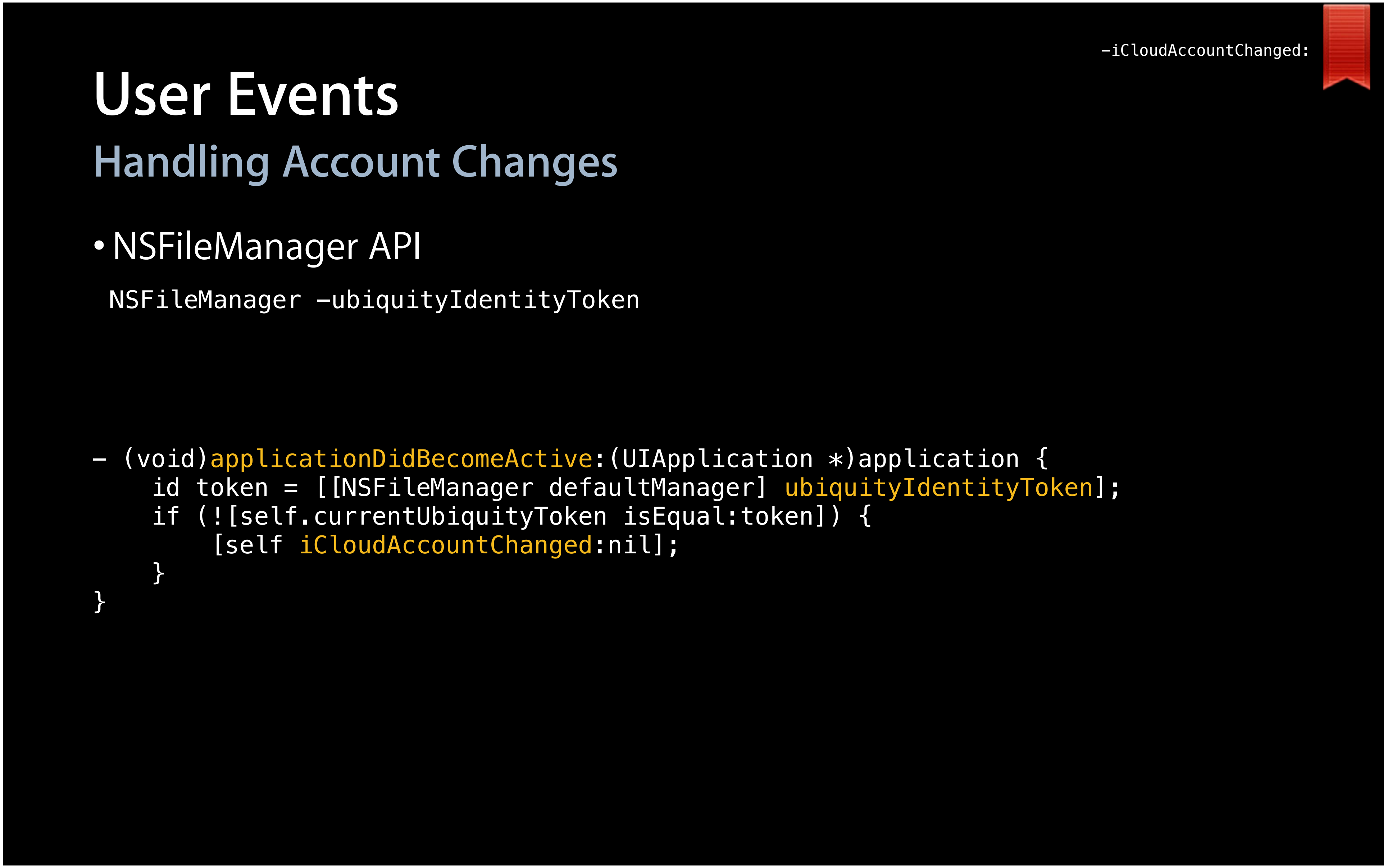
Handling Account Changes

`-iCloudAccountChanged:`

User Events

Handling Account Changes

- NSFileManager API



-iCloudAccountChanged:

User Events

Handling Account Changes

- NSFileManager API

`NSFileManager -ubiquityIdentityToken`

```
- (void)applicationDidBecomeActive:(UIApplication *)application {
    id token = [[NSFileManager defaultManager] ubiquityIdentityToken];
    if (![self.currentUbiquityToken isEqual:token]) {
        [self iCloudAccountChanged:nil];
    }
}
```

-iCloudAccountChanged:

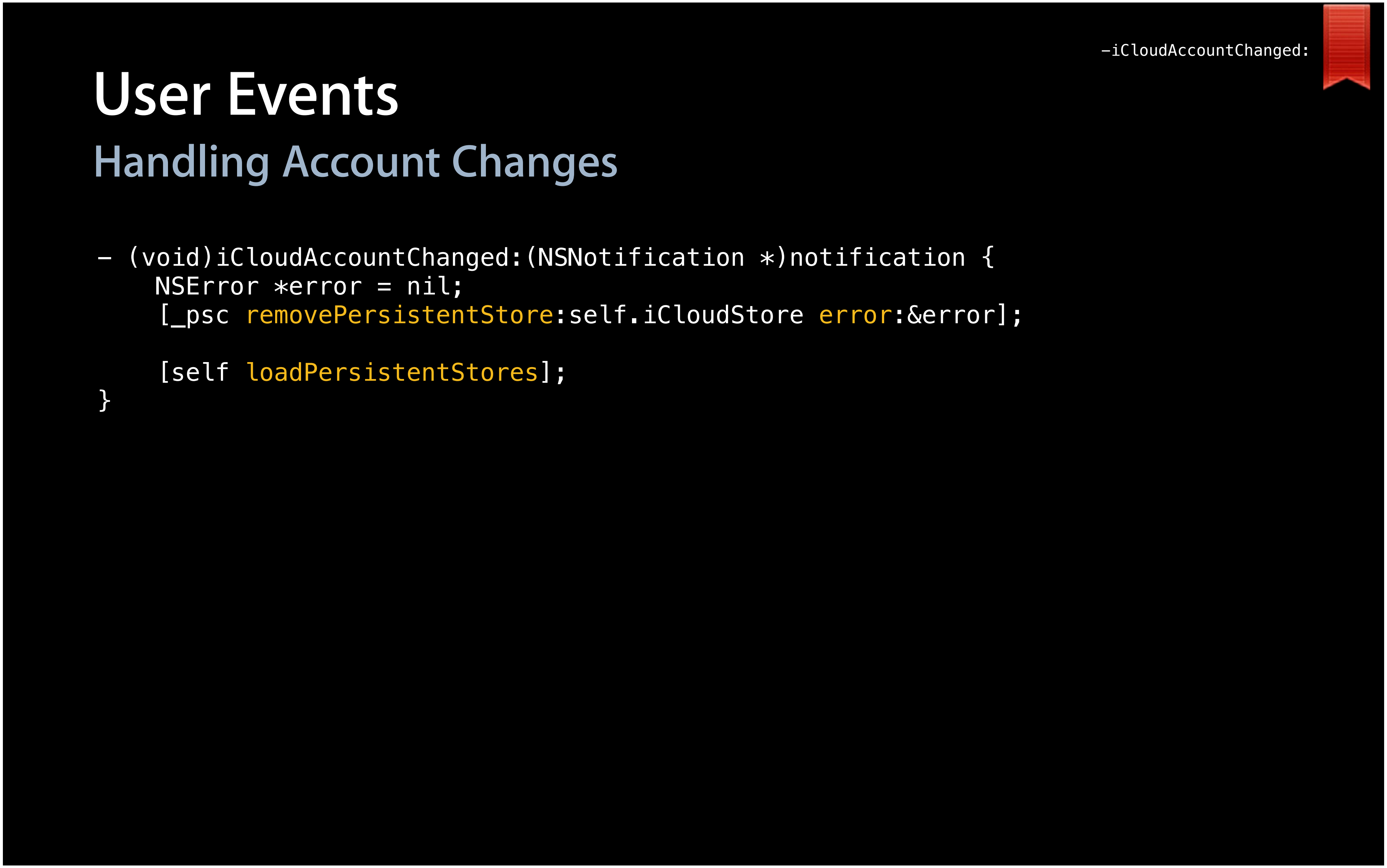
User Events

Handling Account Changes

- NSFileManager API

```
NSFileManager -ubiquityIdentityToken
NSUbiquityIdentityDidChangeNotification
```

```
[notificationCenter addObserver:self
                       selector:@selector(iCloudAccountChanged:)
                           name:NSUbiquityIdentityDidChangeNotification
                         object:nil];
```



-iCloudAccountChanged:

User Events

Handling Account Changes

```
- (void)iCloudAccountChanged:(NSNotification *)notification {
    NSError *error = nil;
    [_psc removePersistentStore:self.iCloudStore error:&error];

    [self loadPersistentStores];
}
```

Demo

User events

Lifecycle

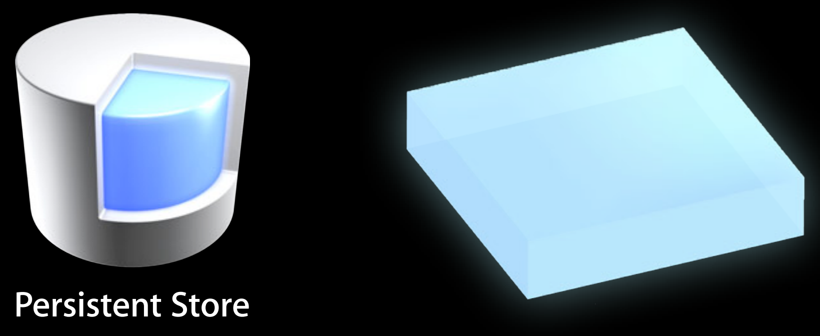
Performance and Debugging

Performance

Performance

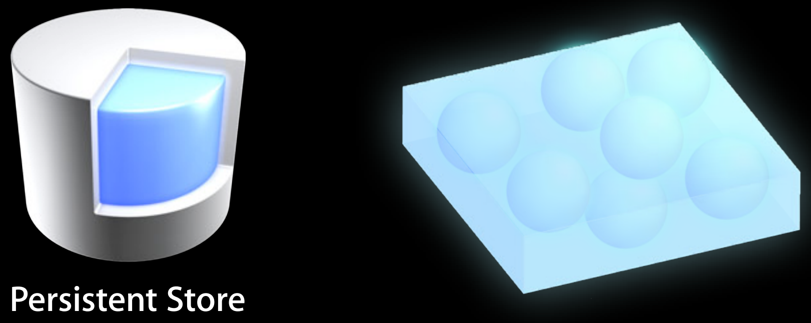
- -save:
- Coalesce changes as appropriate
- Avoid storing raw sensor data
    - CoreLocation @ 60Hz!

Performance



Performance

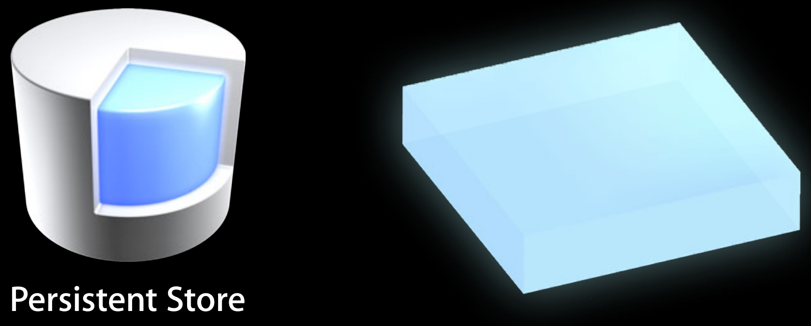
- Memory Pressure



Performance

- Memory Pressure

```
NSFetchRequest -setFetchBatchSize:
NSManagedObjectContext -save:
NSManagedObjectContext -reset
```



Debugging

Debugging

- Macs are greedy peers
    - Run at least one iCloud-enabled app (TextEdit)

Debugging

- Macs are greedy peers
  - Run at least one iCloud-enabled app (TextEdit)
- Removing data
  - Coordinated write to delete every file inside container
  - Be patient, it might take a while to propagate to all your devices

Debugging

- Macs are greedy peers
  - Run at least one iCloud-enabled app (TextEdit)
- Removing data
  - Coordinated write to delete every file inside container
  - Be patient, it might take a while to propagate to all your devices
- File great bugs

Debugging

Filing great bugs

Debugging

Filing great bugs

- Sample Application
- Ubiquity Container
- Console logs
    - -com.apple.coredata.ubiquity.logLevel # (1,2,3)

Demo

Debugging

More Information

Michael Jurewitz
Technology Evangelist
jury@apple.com

Cocoa Feedback
cocoa-feedback@apple.com

Core Data Documentation
Programming Guides, Examples, Tutorials
http://developer.apple.com/

Apple Developer Forums
http://devforums.apple.com

Related Sessions

iCloud Storage Overview	Pacific Heights Tuesday 4:30PM
Core Data Best Practices	Mission Wednesday 9:00AM
Advanced iCloud Document Storage	Marina Thursday 3:15PM

Labs

Core Data Lab	Developer Tools Lab A Thursday 9:00AM
iCloud Storage Lab	Essentials Lab B Thursday 4:30PM
Core Data Lab	Essentials Lab B Friday 9:00AM
iCloud Storage Lab	Essentials Lab B Friday 11:30AM

WWDC2012

The last 3 slides after the logo are intentionally left blank for all presentations.

The last 3 slides after the logo are intentionally left blank for all presentations.

The last 3 slides after the logo are intentionally left blank for all presentations.