What’s New in CloudKit
There's a lot we want to share
Session 226

Paul Seligman CloudKit Engineer
Jacob Farkas CloudKit Engineer
Vanessa Hong CloudKit Engineer
Table of Contents

What are we talking about?
Table of Contents

What are we talking about?

CloudKit Overview
Table of Contents

What are we talking about?

CloudKit Overview
Telemetry
Table of Contents

What are we talking about?

CloudKit Overview
Telemetry
API Improvements
Table of Contents

What are we talking about?

CloudKit Overview
Telemetry
API Improvements
Sharing
CloudKit Overview

What is CloudKit?
CloudKit Overview
Data everywhere!
CloudKit Overview
Data everywhere!

iCloud Database
CloudKit Overview

Data everywhere!

iCloud Database

Extensive use inside Apple
Data everywhere!

CloudKit Overview

iCloud Database
Extensive use inside Apple
Ubiquitous
CloudKit Overview

Prior talks

Introducing CloudKit

WWDC 2014
# CloudKit Overview

Prior talks

<table>
<thead>
<tr>
<th>Introducing CloudKit</th>
<th>WWDC 2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced CloudKit</td>
<td>WWDC 2014</td>
</tr>
<tr>
<td>CloudKit JS and Web Services</td>
<td>WWDC 2015</td>
</tr>
<tr>
<td>What's New in CloudKit</td>
<td>WWDC 2015</td>
</tr>
<tr>
<td>CloudKit Tips and Tricks</td>
<td>WWDC 2015</td>
</tr>
</tbody>
</table>
# CloudKit Overview

## Prior talks

<table>
<thead>
<tr>
<th>Topic</th>
<th>WWDC 2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing CloudKit</td>
<td></td>
</tr>
<tr>
<td>Advanced CloudKit</td>
<td></td>
</tr>
<tr>
<td>CloudKit JS and Web Services</td>
<td></td>
</tr>
<tr>
<td>What's New in CloudKit</td>
<td></td>
</tr>
<tr>
<td>CloudKit Tips and Tricks</td>
<td></td>
</tr>
</tbody>
</table>

https://developer.apple.com/cloudkit
CloudKit Overview

Core objects

Container
Database
Record
Record Zone
CloudKit Overview

Core objects

Container
Database
Record
Record Zone
CloudKit Overview

Core objects

- Container
- Database
- Record
- Record Zone
CloudKit Overview

Core objects

Container
Database
Record
Record Zone

CloudKit Container
CloudKit Overview

Core objects

- Container
- Database
- Record
- Record Zone

CloudKit Container

- Public Database
- Private Database
CloudKit Overview

Core objects

- Container
- Database
- Record
- Record Zone

CloudKit Container

- Public Database
- Private Database
- Shared Database
CloudKit Overview
Core objects

- Container
- Database
- Record
- Record Zone

<table>
<thead>
<tr>
<th>Note</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Value</td>
</tr>
<tr>
<td>Title</td>
<td>String</td>
</tr>
<tr>
<td>Body</td>
<td>String</td>
</tr>
<tr>
<td>CreationDate</td>
<td>Date</td>
</tr>
<tr>
<td>Folder</td>
<td>Reference</td>
</tr>
</tbody>
</table>
CloudKit Overview
Core objects

- Container
- Database
- Record
- Record Zone

Diagram:
- Database
  - Record Zone
Core objects

- Container
- Database
- Record
- Record Zone

CloudKit Overview

Database

Record Zone

Record
CloudKit Overview

Core objects

- Container
- Database
- Record
- Record Zone

Diagram:

- Database
  - Record Zone
    - Record
CloudKit Overview

Core objects

Container
Database
Record
Record Zone
CloudKit Overview

Core objects

- Container
- Database
- Record
- Record Zone

- Public Database
  - Default Zone

- Private Database
  - Default Zone
  - Custom Zone

- Shared Database
  - Shared Zone
  - Custom Zone
  - Shared Zone
CloudKit Overview

Core objects

Container
Database
Record
Record Zone
CloudKit Overview

Core objects

Container
Database
Record
Record Zone
Share
CloudKit Overview

Core objects

Container
Database
Record
Record Zone
Share
CloudKit Overview

Apple usage

- Public Database
- Private Database
- Shared Database

CloudKit Container
CloudKit Overview

Apple usage

CloudKit Container

Public Database

Private Database

Shared Database
CloudKit Overview

Apple usage

CloudKit Container

Public Database

Private Database

Shared Database
CloudKit Overview

Apple usage

CloudKit Container

Public Database

Private Database

Shared Database
CloudKit Is Now Available Everywhere
CloudKit Is Now Available Everywhere
CloudKit Is Now Available Everywhere

macOS  

iOS
CloudKit Is Now Available Everywhere

macOS    iOS

tvOS

CloudKit JS    Web Services
CloudKit Is Now Available Everywhere

macOS  
iOS  
tvOS  
watchOS  
CloudKit JS  
Web Services
macOS
macOS

No Mac App Store requirement
macOS

No Mac App Store requirement

• Add iCloud Capabilities via your provisioning profile
Web

Server to server
Web

Server to server

• Acts as admin user
Web

Server to server

- Acts as admin user
- Uses public/private key pair registered on CloudKit Dashboard
Web

Server to server

• Acts as admin user
• Uses public/private key pair registered on CloudKit Dashboard
• Access to public database
watchOS 3
watchOS 3

Alternative to watch connectivity code
watchOS 3

Alternative to watch connectivity code
Standalone functionality
watchOS 3

Alternative to watch connectivity code

Standalone functionality

• Can work without phone present (via wifi)
watchOS 3

Alternative to watch connectivity code

Standalone functionality

• Can work without phone present (via wifi)

Full* CloudKit API

*offer does not include CKSubscription
watchOS 3

Alternative to watch connectivity code

Standalone functionality

• Can work without phone present (via wifi)

Full* CloudKit API

Similar code on all Apple platforms

*offer does not include CKSubscription
watchOS 3

Alternative to watch connectivity code

Standalone functionality
  • Can work without phone present (via wifi)

Full* CloudKit API

Similar code on all Apple platforms

Limited resources

*offer does not include CKSubscription
Telemetry
Visualize your app's behavior on the CloudKit Dashboard
Telemetry
Visualize your app's behavior on the CloudKit Dashboard
Telemetry

Visualize your app's behavior

CloudKit Dashboard
Telemetry

Visualize your app's behavior

CloudKit Dashboard

Public DB
Telemetry

Visualize your app's behavior

CloudKit Dashboard
Public DB
Aggregated Private DBs
Telemetry
Visualize your app's behavior

CloudKit Dashboard
Public DB
Aggregated Private DBs
Hour/day/week/month
Telemetry
Visualize your app's behavior

CloudKit Dashboard
Public DB
Aggregated Private DBs
Hour/day/week/month
Per Operation type/all Operations
Telemetry

Operations per Second

Average Request Size

Client Errors
Telemetry

Operations per Second

Average Request Size

Client Errors
Performance

Operations per Second

Average Request Size
Correctness

Client Errors

[Graph showing the number of client errors over time]
Correctness

Detect client changes with abnormally frequent errors
Detect client changes with abnormally frequent errors

Error handling is essential
API Improvements
API Improvements
API Improvements

Long Lived Operations
CKOperation Timeouts
Handling Many Record Zones
Fetching Multiple Change Sets
API Improvements

- Long Lived Operations
- CKOperation Timeouts
- Handling Many Record Zones
- Fetching Multiple Change Sets
Long-Lived Operations

Don’t repeat your work
Long-Lived Operations

Don’t repeat your work

Operations keep running if your application exits
Long-Lived Operations

Don’t repeat your work

Operations keep running if your application exits

Callbacks saved by CloudKit
Long-Lived Operations

Don’t repeat your work

Operations keep running if your application exits
Callbacks saved by CloudKit
Operation can be replayed later
API Improvements

Long Lived Operations
CKOperation Timeouts
Handling Many Record Zones
Fetching Multiple Change Sets
API Improvements

Long Lived Operations
CKOperation Timeouts
Handling Many Record Zones
Fetching Multiple Change Sets
# CKOperation Timeouts

How long do you want to wait?

<table>
<thead>
<tr>
<th>QualityOfService</th>
<th>Behavior on broken network</th>
</tr>
</thead>
<tbody>
<tr>
<td>userInteractive</td>
<td></td>
</tr>
<tr>
<td>userInitiated</td>
<td></td>
</tr>
<tr>
<td>utility</td>
<td></td>
</tr>
<tr>
<td>background</td>
<td></td>
</tr>
<tr>
<td>default</td>
<td></td>
</tr>
</tbody>
</table>
## CKOperation Timeouts

How long do you want to wait?

<table>
<thead>
<tr>
<th>QualityOfService</th>
<th>Behavior on broken network</th>
</tr>
</thead>
<tbody>
<tr>
<td>userInteractive</td>
<td>timeout after 60 seconds</td>
</tr>
<tr>
<td>userInitiated</td>
<td>timeout after 60 seconds</td>
</tr>
<tr>
<td>utility</td>
<td></td>
</tr>
<tr>
<td>background</td>
<td></td>
</tr>
<tr>
<td>default</td>
<td></td>
</tr>
</tbody>
</table>
# CKOperation Timeouts

How long do you want to wait?

<table>
<thead>
<tr>
<th>QualityOfService</th>
<th>Behavior on broken network</th>
</tr>
</thead>
<tbody>
<tr>
<td>userInteractive</td>
<td>timeout after 60 seconds</td>
</tr>
<tr>
<td>userInitiated</td>
<td>timeout after 60 seconds</td>
</tr>
<tr>
<td>utility</td>
<td>timeout after 7 days</td>
</tr>
<tr>
<td>background</td>
<td>timeout after 7 days</td>
</tr>
<tr>
<td>default</td>
<td>timeout after 7 days</td>
</tr>
</tbody>
</table>
## CKOperation Timeout

How long do you want to wait?

<table>
<thead>
<tr>
<th>QualityOfService</th>
<th>Behavior on broken network</th>
</tr>
</thead>
<tbody>
<tr>
<td>userInteractive</td>
<td>timeout after 60 seconds</td>
</tr>
<tr>
<td>userInitiated</td>
<td>timeout after 60 seconds</td>
</tr>
<tr>
<td>utility</td>
<td>timeout after 7 days</td>
</tr>
<tr>
<td>background</td>
<td>timeout after 7 days</td>
</tr>
<tr>
<td>default</td>
<td>timeout after 7 days</td>
</tr>
</tbody>
</table>
# CKOperation Timeouts

How long do you want to wait?

<table>
<thead>
<tr>
<th>QualityOfService</th>
<th>Behavior on broken network</th>
</tr>
</thead>
<tbody>
<tr>
<td>userInteractive</td>
<td>timeout after 60 seconds</td>
</tr>
<tr>
<td>userInitiated</td>
<td>timeout after 60 seconds</td>
</tr>
<tr>
<td>utility</td>
<td>timeout after 7 days</td>
</tr>
<tr>
<td>background</td>
<td>timeout after 7 days</td>
</tr>
<tr>
<td>default</td>
<td>timeout after 7 days</td>
</tr>
</tbody>
</table>
CKOperation Timeouts
How long do you want to wait?
CKOperation Timeouts

How long do you want to wait?

Network inactivity

• Use the `timeoutIntervalForRequest` property on CKOperation
• Default value is 60 seconds
CKOperation Timeouts
How long do you want to wait?

Network inactivity
- Use the `timeoutIntervalForRequest` property on CKOperation
- Default value is 60 seconds

Start-to-finish timeout
- Use the `timeoutIntervalForResource` property on CKOperation
- Default value is 7 days
- CKOperation may stay around longer
API Improvements

Long Lived Operations
CKOperation Timeouts
Handling Many Record Zones
Fetching Multiple Change Sets
API Improvements

Long Lived Operations
CKOperation Timeouts
Handling Many Record Zones
Fetching Multiple Change Sets
Handling Many Record Zones

Reduce payloads and roundtrips

CKFetchRecordZonesOperation

- Poll for all record zones in a database
Handling Many Record Zones
Reduce payloads and roundtrips

CKFetchRecordZonesOperation
- Poll for all record zones in a database
Handling Many Record Zones
Reduce payloads and roundtrips

CKFetchRecordZonesOperation
- Poll for all record zones in a database
Handling Many Record Zones
Reduce payloads and roundtrips

CKDatabaseSubscription
- Receive a push for each change in a database
CKFetchDatabaseChangesOperation
- Fetch ids of record zones with changes

CKFetchRecordZonesOperation
- Poll for all record zones in a database
Handling Many Record Zones
Reduce payloads and roundtrips

CKDatabaseSubscription
• Receive a push for each change in a database

CKFetchDatabaseChangesOperation
• Fetch ids of record zones with changes

CKFetchRecordChangesOperation
• Track and fetch record changes on a per-record-zone basis

CKFetchRecordZonesOperation
• Poll for all record zones in a database
Handling Many Record Zones
Reduce payloads and roundtrips

- CKDatabaseSubscription
  - Receive a push for each change in a database

- CKFetchDatabaseChangesOperation
  - Fetch ids of record zones with changes

- CKFetchRecordChangesOperation
  - Track and fetch record changes on a per-record-zone basis

- CKFetchRecordZonesOperation
  - Poll for all record zones in a database
Handling Many Record Zones
Reduce payloads and roundtrips

CKDatabaseSubscription
- Receive a push for each change in a database
- Fetch ids of record zones with changes

CKFetchDatabaseChangesOperation
- Poll for all record zones in a database

CKFetchRecordZonesOperation
- Track and fetch record changes on a per-record-zone basis
Handling Many Record Zones
Reduce payloads and roundtrips

- **CKDatabaseSubscription**
  - Receive a push for each change in a database

- **CKFetchDatabaseChangesOperation**
  - Fetch ids of record zones with changes

- **CKFetchRecordZoneChangesOperation**
  - Fetch record changes over multiple record zones in a single operation

- **CKFetchRecordChangesOperation**
  - Track and fetch record changes on a per-record-zone basis

- **CKFetchRecordZonesOperation**
  - Poll for all record zones in a database
Handling Many Record Zones

CKDatabaseSubscription
Handling Many Record Zones

CKDatabaseSubscription

Database

Record Zone

Record

Record

Record

Record
Handling Many Record Zones

CKDatabaseSubscription
Handling Many Record Zones

CKDatabaseSubscription

Database

- Record Zone
- Record
- Record
- Record
- Record
Handling Many Record Zones

CKFetchDatabaseChangesOperation
Handling Many Record Zones

CKFetchDatabaseChangesOperation
Handling Many Record Zones
CKFetchRecordZoneChangesOperation
Handling Many Record Zones

CKFetchRecordZoneChangesOperation
API Improvements

Long Lived Operations
CKOperation Timeouts
Handling Many Record Zones
Fetching Multiple Change Sets
API Improvements

Long Lived Operations
CKOperation Timeouts
Handling Many Record Zones
Fetching Multiple Change Sets
Fetch Multiple Change Sets
Remember this?

```swift
public class CKFetchRecordChangesOperation : CKDatabaseOperation {
    public var moreComing: Bool { get }
}
```
Fetch Multiple Change Sets

Remember this?

```swift
public class CKFetchRecordChangesOperation : CKDatabaseOperation {
    public var moreComing: Bool { get }
}
```

Client code responsible for fetching next batch
Fetch Multiple Change Sets

Remember this?

```swift
public class CKFetchRecordChangesOperation : CKDatabaseOperation {
    public var moreComing: Bool { get }
}
```

Client code responsible for fetching next batch

CloudKit idle
Fetch Multiple Change Sets

Use this instead

```swift
public class CKFetchRecordZoneChangesOperation : CKDatabaseOperation {

    public var fetchAllChanges: Bool

}
```
Fetch Multiple Change Sets

Use this instead

```swift
public class CKFetchRecordZoneChangesOperation : CKDatabaseOperation {
    public var fetchAllChanges: Bool
}
```

CloudKit keeps pipeline full
Fetch Multiple Change Sets
Use this instead

```swift
public class CKFetchRecordZoneChangesOperation : CKDatabaseOperation {
    // fetchAllChanges is true by default
    public var fetchAllChanges: Bool
}
```

CloudKit keeps pipeline full
Fetch Multiple Change Sets

New callback

```swift
public class CKFetchRecordZoneChangesOperation : CKDatabaseOperation {
    public var recordZoneChangeTokensUpdatedBlock: ((CKRecordZoneID, CKServerChangeToken?, Data?) -> Void)?
}
```
Fetch Multiple Change Sets

New callback

```swift
public class CKFetchRecordZoneChangesOperation : CKDatabaseOperation {
    public var recordZoneChangeTokensUpdatedBlock: ((CKRecordZoneID, CKServerChangeToken?, Data?) -> Void)?
}
```

Earlier record changes are safe to commit
Fetch Multiple Change Sets

New callback

public class CKFetchRecordZoneChangesOperation : CKDatabaseOperation {
    public var recordZoneChangeTokensUpdatedBlock: ((CKRecordZoneID, CKServerChangeToken?, Data?) -> Void)?
}

Earlier record changes are safe to commit

New server change token can be used on a new CKFetchRecordZoneChangesOperation
API Improvements

- Long Lived Operations
- CKOperation Timeouts
- Handling Many Record Zones
- Fetching Multiple Change Sets
Sharing UI
You and I are going to share some records

Jacob Farkas
Sharing Records

public class CKShare : CKRecord
Sharing Records

What is shared?

public class CKShare : CKRecord
Sharing Records

What is shared?
Who is it shared with?

```java
public class CKShare : CKRecord
```
Sharing Overview

Shared database

CloudKit Container

Public Database

Private Database

Shared Database
Sharing Records

What is shared?

Private Database
Sharing Records

What is shared?

Private Database

Note
Sharing Records

What is shared?

Private Database

Note
Sharing Records

What is shared?

Private Database

Share

Note

public class CKShare : CKRecord {
    public init(rootRecord : CKRecord)
}
Sharing Records

What is shared?

```
public class CKShare : CKRecord {
    public init(rootRecord : CKRecord)
}]
```
Sharing Records
What is shared?

```swift
public class CKShare : CKRecord {
    public init(rootRecord : CKRecord)
}
```

```swift
public class CKRecord {
    public var share: CKReference? { get }
}
```
Sharing Records

Who is it shared with?

Private Database

Share

Note
Sharing Records

Who is it shared with?

Private Database

Share

Note
Sharing Records
Who is it shared with?

Private Database
Share
Note

user@icloud.com
Sharing Records

Who is it shared with?

Private Database

Share

Note
Sharing Records

Who is it shared with?

Private Database

Share

Note
Sharing Records

Who is it shared with?

Private Database

Share

Note

Cloud Icon
Sharing Records

Who is it shared with?

Private Database

Share

Note
Sharing Records

Share URLs

Private Database

Share

Note
Sharing Records

Share URLs

https://www.icloud.com/notes/000Y4qow0owP6N0xDzs4qgi8Q
Sharing Records

Share URLs

Private Database

Share

Note

Derek Parker
“Family Grocery list”
To: Emily Parker

Open my shared note:

“Family Grocery list”
icloud.com/notes
Sharing Records

Accepting a share

Owner

Private Database

Share

Note

Other User
Sharing Records

Accepting a share

Owner

Private Database

Share

Note

Other User
Sharing Records
Accepting a share

Owner

Private Database
Share
Note

Other User
Sharing Records
Accepting a share

Owner

Private Database
Share
Note

Other User

Shared Database
Share
Note
Sharing Records

Owner

Private Database
Share
Note

Other User

Shared Database
Share
Note
Sharing Records

Owner

Private Database

Share

Note

Other User

Shared Database

Share

Note
Demo
System Sharing UI
CloudKit Sharing UI

Where does it live?
CloudKit Sharing UI

Where does it live?

CloudKit

CKRecord
CKShare
CKModifyRecordsOperation

...
CloudKit Sharing UI
Where does it live?

CloudKit
- CKRecord
- CKShare
- CKModifyRecordsOperation
...

macOS
- AppKit
- NSSharingService
- NSItemProvider

iOS
- UIKit
- UICloudSharingController
iOS Sharing API

UICloudSharingController
// Create a CloudKit share record
let share = CKShare(rootRecord: rootRecord)
share[CKShareTitleKey] = "Shopping List"
share[CKShareThumbnailImageDataKey] = shoppingListThumbnail
// Create a CloudKit share record

let share = CKShare(rootRecord: rootRecord)
share[CKShareTitleKey] = "Shopping List"
share[CKShareThumbnailImageDataKey] = shoppingListThumbnail
// Create a CloudKit share record

let share = CKShare(rootRecord: rootRecord)

share[CKShareTitleKey] = "Shopping List"

share[CKShareThumbnailImageDataKey] = shoppingListThumbnail
// Create a CloudKit share record

let share = CKShare(rootRecord: rootRecord)
share[CKShareTitleKey] = "Shopping List"
share[CKShareThumbnailImageDataKey] = shoppingListThumbnail
// Create a cloud sharing controller

let sharingController = UFCloudSharingController(share: share) {
    (controller: UFCloudSharingController,
     prepareCompletionHandler : (CKShare?, CKContainer?, NSError?) -> Void) in
// Create a cloud sharing controller

let sharingController = UICloudSharingController(share: share) {
    (controller: UICloudSharingController,
     prepareCompletionHandler: (CKShare?, CKContainer?, NSError?) -> Void) in
// Save the share

let sharingController = UICloudSharingController(share: share) {
    (controller: UICloudSharingController,
     prepareCompletionHandler : (CKShare?, CKContainer?, NSError?) -> Void) in

    let modifyOp = CKModifyRecordsOperation(recordsToSave: [rootRecord, share],
                                             recordIDsToDelete: nil)
    modifyOp.modifyRecordsCompletionBlock = { (_, _, error) in
                                             prepareCompletionHandler(share, ckContainer, error)
                                        }

    self.container.privateCloudDatabase.add(modifyOp)
}
// Save the share

let sharingController = UICloudSharingController(share: share) {
    (controller: UICloudSharingController,
     prepareCompletionHandler : (CKShare?, CKContainer?, NSError?) -> Void) in

    let modifyOp = CKModifyRecordsOperation(recordsToSave: [rootRecord, share],
                                             recordIDsToDelete: nil)
    modifyOp.modifyRecordsCompletionBlock = { (_, _, error) in
        prepareCompletionHandler(share, ckContainer, error)
    }
    self.container.privateCloudDatabase.add(modifyOp)
}
// Save the share

let sharingController = UICloudSharingController(share: share) {
    (controller: UICloudSharingController,
     prepareCompletionHandler : (CKShare?, CKContainer?, NSError?) -> Void) in

    let modifyOp = CKModifyRecordsOperation(recordsToSave: [rootRecord, share],
                                               recordIDsToDelete: nil)

    modifyOp.modifyRecordsCompletionBlock = { (_, _, error) in

        prepareCompletionHandler(share, ckContainer, error)

    }

    self.container.privateCloudDatabase.add(modifyOp)
}

// Set sharing options

sharingController.availablePermissions = [.publicOnly, .readWrite]
sharingController.popoverPresentationController?.sourceView = myShareButton
sharingController.delegate = self
self.present(sharingController, animated: true, completion: nil)
// Set sharing options

sharingController.availablePermissions = [.publicOnly, .readWrite]
sharingController.popoverPresentationController?.sourceView = myShareButton
sharingController.delegate = self
self.present(sharingController, animated: true, completion: nil)
/ Set sharing options

sharingController.availablePermissions = [.publicOnly, .readWrite]

sharingController.popoverPresentationController?.sourceView = myShareButton

sharingController.delegate = self

self.present(sharingController, animated: true, completion: nil)
// Set sharing options

sharingController.availablePermissions = [.publicOnly, .readWrite]
sharingController.popoverPresentationController?.sourceView = myShareButton
sharingController.delegate = self

self.present(sharingController, animated:true, completion:nil)
// Set sharing options

sharingController.availablePermissions = [.publicOnly, .readWrite]
sharingController.popoverPresentationController?.sourceView = myShareButton
sharingController.delegate = self

self.present(sharingController, animated: true, completion: nil)
Family Grocery list

Dad is going to Trader Joe’s on Thursday. Don’t forget the BBQ this Saturday. Also:

- Milk - gallon
- Butter
- Eggs - two dozen
- 5 racks baby back ribs
- Avocados (6)
- Corn chips
- Tomato sauce

Add People

Share this with others and everyone will see the latest changes.

Family Grocery list
Derek Parker (derek_wdcd16@icloud.com)

Choose how you’d like to send your invitation:
- Message
- Mail
- Copy Link
- Twitter

People you invite can make changes
Family Grocery list

Dad is going to Trader Joe’s on Thursday.

Don’t forget the bbq this Saturday. Add:

- Milk – gallon
- Butter
- Eggs – two dozen
- 5 racks baby back ribs
- Avocados (6)
- Corn chips
- Tomato sauce

Choose how you’d like to send your invitation:
- Message
- Mail
- Copy Link
- Twitter
macOS Sharing API
NSSharingService
// Save the share
let itemProvider = NSItemProvider()
itemProvider.registerCloudKitShare { (prepareCompletionHandler :
    (CKShare?, CKContainer?, NSError?) -> Void) in
    // Save the share and root record
}

let sharingService = NSSharingService(named: NSSharingServiceNameCloudSharing)!
sharingService.delegate = self
sharingService.perform(withItems: [itemProvider])
// Save the share

let itemProvider = NSItemProvider()
itemProvider.registerCloudKitShare { (prepareCompletionHandler : (CKShare?, CKContainer?, NSError?) -> Void) in
    // Save the share and root record
}

let sharingService = NSSharingService(named: NSSharingServiceNameCloudSharing)!
sharingService.delegate = self
sharingService.perform(withItems: [itemProvider])
// Save the share
let itemProvider = NSItemProvider()
itemProvider.registerCloudKitShare { (prepareCompletionHandler: (CKShare?, CKContainer?, NSError?) -> Void) in
    // Save the share and root record
}

let sharingService = NSSharingService(named: NSSharingServiceNameCloudSharing)!
sharingService.delegate = self
sharingService.perform(withItems: [itemProvider])
// Save the share
let itemProvider = NSItemProvider()
itemProvider.registerCloudKitShare { (prepareCompletionHandler : (CKShare?, CKContainer?, NSError?) -> Void) in
    // Save the share and root record
}
let sharingService = NSSharingService(named: NSSharingServiceNameCloudSharing)!
sharingService.delegate = self
sharingService.perform(withItems: [itemProvider])
// Save the share
let itemProvider = NSItemProvider()
itemProvider.registerCloudKitShare { (prepareCompletionHandler: (CKShare?, CKContainer?, NSError?) -> Void) in
  // Save the share and root record
}

let sharingService = NSSharingService(named: NSSharingServiceNameCloudSharing)!
sharingService.delegate = self
sharingService.perform(withItems: [itemProvider])
// Define sharing options

```
func options(for: NSSharingService, share: NSItemProvider) -> NSCloudKitSharingServiceOptions {
    return [.allowPublic, .allowReadWrite]
}
```
// Define sharing options

func options(for: NSSharingService, share: NSItemProvider) -> NSCloudKitSharingServiceOptions {
    return [.allowPublic, .allowReadWrite]
}
Add People
Choose how you'd like to send your invitation:

- Mail
- Messages
- Copy Link
- Twitter
- Facebook
- AirDrop

Only people you invite can make changes.

Derek Parker (derek_wwdc16@icloud.com)  Cancel  Share
People
Only people you invite can make changes.

Derek Parker (Owner)

Emily Parker
Invited

Add People

Copy Link  Stop Sharing
// User clicked a share

public class NSApplication {
    public func application(application: NSApplication,
                            userAcceptedCloudKitShareWith: CKShareMetadata)
}

macOS
public class NSApplication {
    public func application(application: NSApplication,
        userAcceptedCloudKitShareWith: CKShareMetadata)
}

public class UIApplication {
    public func application(application: UIApplication,
        userAcceptedCloudKitShareWith: CKShareMetadata)
}
// Add an Info.plist key for CloudKit Sharing

<key>CKSharingSupported</key>
<true/>
CloudKit JS
Web Sharing UI
.shareWithUI()

This sample shows how to share a record with the default sharing UI.

```javascript
var container = CloudKit.getDefaultContainer();
var database = container.getDefaultDatabaseWithDatabaseScope(  
    CloudKit.DatabaseScope[databaseScope]  
);

var zoneID = { zoneName: zoneName };  
if(ownerRecordName) {
    zoneID.ownerRecordName = ownerRecordName;
}

return database.shareWithUI({
    record: {
```
Sharing In Depth

Vanessa Hong
Common Use Cases
Deep dive

Sharing multiple records
Zones in shared database
CKShare internals
Sharing APIs
Special notes
Sharing Multiple Records

A Note is not a single record
A Note consists of many records.
Sharing Multiple Records

A Note consists of many records

Diagram:
- Note
  - Media
    - Asset
  - Links
    - Data
Sharing Multiple Records
A Note consists of many records
Sharing Multiple Records
Participant should only see a subset
Sharing Multiple Records
Tell us what should be shared

Records have a new property

public var parent: CKReference
Sharing Multiple Records

Parent references define the hierarchy for sharing

Records have a new property

public var parent: CKReference
Sharing Multiple Records

Descendants linked to root record via the parent field

Records have a new property

public var parent: CKReference
Sharing Multiple Records
Create Share using CKShare(rootRecord:)

Records have a new property

public var parent: CKReference
Sharing Multiple Records

Shared DB is only a View into the owner’s private DB
Sharing Multiple Records

After accepting, Participant sees only records with parent
Sharing Multiple Records

After accepting, Participant sees only records with parent
Sharing Multiple Records

After accepting, Participant sees only records with parent

Private Database

Shared Database

Owner

Participant
Sharing Multiple Records

readWrite Participant cannot add a dangling CKRecord
Sharing Multiple Records

ReadWrite Participant cannot add a dangling CKRecord
Sharing Multiple Records

readWrite Participant cannot add a dangling CKRecord
Sharing Multiple Records

readWrite Participant cannot add a dangling CKRecord
Sharing Multiple Records
readWrite Participant cannot add a dangling CKRecord

Private Database

Shared Database
Error: No Parent
Sharing Multiple Records

readWrite Participant can add a new parented CKRecord

Owner

Private Database

Shared Database

Participant
Sharing Multiple Records

**readWrite Participant** can add a new parented CKRecord

**Private Database**

**Shared Database**
Zones in Shared Database
Two owners, two shares
Zones in Shared Database
Two owners, two shares, two zones

Shared Database

Record Zone

“Shopping List” → Share

Record Zone

“Shopping List” → Share
Zones in Shared Database

Two owners, two shares, two zones

public class CKRecordZoneID {
    public var zoneName: String
    public var ownerName: String
}
Zones in Shared Database

Two owners, two shares, two zones

public class CKRecordZoneID {
    public var zoneName: String
    public var ownerName: String
}

Shared Database

zoneName: “Notes”

“Shopping List” → Share

zoneName: “Notes”

“Shopping List” → Share
Zones in Shared Database
Two owners, two shares, two zones

public class CKRecordZoneID {
    public var zoneName: String
    public var ownerName: String
}

Shared Database

zoneName: “Notes”

“Shopping List” → Share

zoneName: “Notes”

“Shopping List” → Share
Zones in Shared Database
Two owners, two shares, two zones

public class CKRecordZoneID {
    public var zoneName: String
    public var ownerName: String
}

zoneName: “Notes”
ownerName: “_abcxyz”

zoneName: “Notes”
ownerName: “_1234567”

“Shopping List” → Share
“Shopping List” → Share
Zones in Shared Database

Two owners, two shares, two zones

public class CKRecordZoneID {
    public var zoneName: String
    public var ownerName: String
}

Shopping List

zoneName: "Notes"
ownerName: "_abcxyz"

zoneName: "Notes"
ownerName: "_1234567"
Zones in Shared Database

Two owners, three shares, three zones

Shared Database

zoneName: "Notes"
ownerName: "_abcxyz"

"Shopping List" → Share

zoneName: "Notes"
ownerName: "_1234567"

"Shopping List" → Share

zoneName: "OtherZone"
ownerName: "_abcxyz"

"Recipe" → Share
Zones in Shared Database
Two owners, three shares, three zones

- Zone 1: zoneName: "Notes", ownerName: "_abcxyz"
  - "Shopping List"
  - "Recipe"

- Zone 2: zoneName: "Notes", ownerName: "_1234567"
  - "Shopping List"

- Zone 3: zoneName: "OtherZone", ownerName: "_abcxyz"
  - Share
Zones in Shared Database
Two owners, four shares, three zones

Shared Database

- zoneName: "Notes"  
  ownerName: "_abcxyz"
  "Shopping List"  
  "Recipe"

- zoneName: "Notes"  
  ownerName: "_1234567"
  "Shopping List"
  "Great Hiking Trails"
Two owners, four shares, three zones

Shared Database

- zoneName: "Notes"  
  ownerName: "_abcxyz"
  "Shopping List" ➔ Share
  "Recipe" ➔ Share

- zoneName: "Notes"  
  ownerName: "_1234567"
  "Shopping List" ➔ Share
  "Great Hiking Trails" ➔ Share
Prerequisite—Owner has existing Record(s) to share
CKShare

Prerequisite—Owner has existing Record(s) to share

“What Shopping List”

What to Share
CKShare
Prerequisite—Owner has existing Record(s) to share

What to Share

“Shopping List”

How to Share

Share
CKShare

Is a CKRecord containing access controls for shared data

Every CKShare has additional properties beyond a basic CKRecord
**CKShare**

Is a CKRecord containing access controls for shared data

Every CKShare has additional properties beyond a basic CKRecord

```swift
public class CKShare : CKRecord {
    public var participants: [CKShareParticipant]
}
```

```swift
public class CKShareParticipant
```
CKShare Lifecycle—Invite-Only
Owner sets up the Share
CKShare Lifecycle—Invite-Only

Owner sets up the Share

1. Create a Share
CKShare Lifecycle—Invite-Only
Owner sets up the Share

1. Create a Share
2. publicPermission=None

publicPermission: none
CKShare Lifecycle—Invite-Only

Owner sets up the Share

1. Create a Share
2. publicPermission=none
3. Add Participant
   1. acceptanceStatus=invited
   2. Owner determines each participant’s permission

Share

publicPermission: none
participants:

Participant #1
  acceptanceStatus: invited
  permission: readWrite

Participant #2
  acceptanceStatus: invited
  permission: readOnly
<table>
<thead>
<tr>
<th>Participant</th>
<th>Acceptance Status</th>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Invited</td>
<td>readWrite</td>
</tr>
<tr>
<td>#2</td>
<td>Invited</td>
<td>readOnly</td>
</tr>
</tbody>
</table>

1. Create a Share
2. publicPermission = none
3. Add Participant
4. Owner determines each participant's permission
5. Save the Share
**CKShare Lifecycle—Invite-Only**

Owner sets up the Share

1. Create a Share
2. `publicPermission=none`
3. Add Participant
   1. `acceptanceStatus=invited`
   2. Owner determines each participant’s permission
4. Save the Share
5. Owner gets URL

<table>
<thead>
<tr>
<th>Participant #1</th>
<th>Acceptance Status: invited</th>
<th>Permission: readWrite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participant #2</td>
<td>Acceptance Status: invited</td>
<td>Permission: readOnly</td>
</tr>
</tbody>
</table>
CKShare Lifecycle—Invite-Only

Participant joins the Share

Share

url: “Shopping List” icloud.com/notes
publicPermission: none
participants:

Participant #1
  acceptanceStatus: invited
  permission: readWrite

Participant #2
  acceptanceStatus: invited
  permission: readOnly
1. Participants accept via URL
   acceptanceStatus=accepted
Owner sets up the Share

CKShare Lifecycle—readOnly or readWrite
CKShare Lifecycle—readOnly or readWrite

Owner sets up the Share

1. Create a Share
CKShare Lifecycle—readOnly or readWrite

Owner sets up the Share

1. Create a Share
2. publicPermission=readOnly or readWrite
CKShare Lifecycle—*readOnly* or *readWrite*

Owner sets up the Share

1. Create a Share
2. `publicPermission=readOnly` or *readWrite*
3. Save the Share
CKShare Lifecycle—readOnly or readWrite

Owner sets up the Share

1. Create a Share
2. publicPermission=readOnly
   or readWrite
3. Save the Share
4. Owner gets URL

url: "Shopping List"
icloud.com/notes

publicPermission: readOnly or readWrite
CKShare Lifecycle—readOnly or readWrite
Participant joins the Share

1. Anyone can join via URL. 
   acceptanceStatus=accepted, 
   permission is the same as the 
   publicPermission 

url: "Shopping List"  
url: icloud.com/notes 
publicPermission: readOnly or readWrite 
participants: 
Participant #1 
acceptanceStatus: accepted
Participant leaves the Share by deleting the Share from their Shared DB
CKShare

Lifecycle—end the share for a participant

Participant leaves the Share by deleting the Share from their Shared DB
CKShare

Lifecycle—end the share for a participant

Participant leaves the Share by deleting the Share from their Shared DB

Owner can always remove any Participant
CKShare

Lifecycle—End the share for everyone

Owner deletes the Share from his private DB
CKShare
Lifecycle—End the share for everyone

Owner deletes the Share from his private DB
public class CKShareParticipant {
    public var userIdentity: CKUserIdentity
    ...
}

public class CKUserIdentity {
    public var lookupInfo: CKUserIdentityLookupInfo
    public var nameComponents: PersonNameComponents
}

CKShareParticipant
CKUserIdentity
public class CKShareParticipant {
    public var userIdentity: CKUserIdentity
    ...
}

public class CKUserIdentity {
    public var lookupInfo: CKUserIdentityLookupInfo
    public var nameComponents: PersonNameComponents
}
public class CKShareParticipant {
    public var userIdentity: CKUserIdentity
    ...
}

public class CKUserIdentity {
    public var lookupInfo: CKUserIdentityLookupInfo
    public var nameComponents: PersonNameComponents
}

public class CKShareParticipant {
    public var userIdentity: CKUserIdentity
    ...
}

public class CKUserIdentity {
    public var lookupInfo: CKUserIdentityLookupInfo
    public var nameComponents: PersonNameComponents
}
CKShareParticipant
Mapped to iCloud accounts
CKShareParticipant
Mapped to iCloud accounts

Participant #1
userIdentity.lookupInfo: <email>

Participant #2
userIdentity.lookupInfo: <phone>

Participant #3
userIdentity.lookupInfo: <email>

Participant #4
userIdentity.lookupInfo: <phone>
CKShareParticipant
Mapped to iCloud accounts

Participant #1
userIdentity.lookupInfo: <email>

Participant #2
userIdentity.lookupInfo: <phone>

Participant #3
userIdentity.lookupInfo: <email>

Participant #4
userIdentity.lookupInfo: <phone>
## CKShareParticipant

**Mapped to iCloud accounts**

<table>
<thead>
<tr>
<th>Participant #1</th>
<th>userIdentity.lookupInfo: &lt;email&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participant #2</td>
<td>userIdentity.lookupInfo: &lt;phone&gt;</td>
</tr>
<tr>
<td>Participant #3</td>
<td>No iCloud account</td>
</tr>
<tr>
<td></td>
<td>Verification flow to prove email ownership</td>
</tr>
<tr>
<td>Participant #4</td>
<td>No iCloud account</td>
</tr>
<tr>
<td></td>
<td>Verification flow to prove phone ownership</td>
</tr>
</tbody>
</table>
Sharing APIs

If you want to create your own Custom UI

On behalf of the owner

• Setting up the Share

On behalf of the participant

• Accept the Share

watchOS and tvOS

• Shared records available, but no System UI
Owner Sets Up the Share

Adding participants
Owner Sets Up the Share

Adding participants

CKFetchShareParticipantsOperation

• Can look up via
  - Email
  - Phone
  - CloudKit User Record ID
Owner Sets Up the Share

Adding participants

CKFetchShareParticipantsOperation

• Can look up via
  - Email
  - Phone
  - CloudKit User Record ID

• Returns CKShareParticipants
Owner Sets Up the Share

Adding participants

CKFetchShareParticipantsOperation

- Can look up via
  - Email
  - Phone
  - CloudKit User Record ID

- Returns CKShareParticipants

Pass CKShareParticipants to addParticipant
Owner Sets Up the Share

Adding participants

CKFetchShareParticipantsOperation

- Can look up via
  - Email
  - Phone
  - CloudKit User Record ID

- Returns CKShareParticipants

Pass CKShareParticipants to addParticipant

Call CKModifyRecordsOperation to save the share
Participant Accepts a Share

Fetch Share Metadata, then Accept the Share

CKFetchShareMetadataOperation

• Converting a URL to CKShareMetadata

Pass CKShareMetadata to CKAcceptSharesOperation
Participant Accepts a Share

Limitations

No `nameComponents`, for privacy reasons

```java
public class CKUserIdentity {
    public var nameComponents: PersonNameComponents // empty
}
```

Verification flow only available via System UI:

- `CKErrorParticipantMayNeedVerification`
- `shareParticipant.userIdentity.lookupInfo.hasiCloudAccount`
Sharing

Invitees on older platforms

Owner can invite anybody

- Invitee may not have installed the latest operating system
- Invitee may not have an Apple product
"Grocery List"

Emily Parker shared this note. You will join as Derek Parker (derek.parker@icloud.com).

Open in Notes
Not Now
Grocery List

- Milk (1 gallon, 2%)
- Eggs (dozen)
- Quinoa
- Peaches
- Raspberries
- Goldfish
- Almonds

Dogs are cool.
Yesterday Yup. They are.
Sharing

Get your container ready for Sharing

CKRecordTypeShare
• Behaves like any other Record Type in CloudKit
• Can create custom fields
• Can run queries

To trigger its creation
• Share a record in a custom zone in any private database in the development environment
• Deploy schema to production

Now, users in production can create Shares
Summary

What’s new is now old

CloudKit is available on all platforms
Telemetry available on CloudKit Dashboard
API Improvements
New Feature—Sharing
• Sharing System UI
• Sharing APIs, Objects, and Lifecycle
• Configure your fallback URL!
More Information

## Related Sessions

<table>
<thead>
<tr>
<th>CloudKit Best Practices</th>
<th>Pacific Heights</th>
<th>Friday 9:00AM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labs</td>
<td>Frameworks Lab D</td>
<td>Thursday 4:00PM</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>CloudKit and iCloud Lab</td>
<td>Frameworks Lab B</td>
<td>Friday 12:00PM</td>
</tr>
</tbody>
</table>