System Frameworks

#WWDC15

CloudKit Tips And Tricks

Session 715

Nihar Sharma CloudKit Engineer

CloudKit

What is it?

CloudKit

What is it?

iCloud database

CloudKit

What is it?

iCloud database

Large file storage

CloudKit

What is it?

iCloud database

Large file storage

Privacy obsessed

CloudKit

What is it?

iCloud database

Large file storage

Privacy obsessed

Developer API

CloudKit

What is it?

iCloud database

Large file storage

Privacy obsessed

Developer API

Apple applications built on it

CloudKit

One year later

CloudKit

No really, what is it?

Introduction to CloudKit	WWDC14
Advanced CloudKit	WWDC14

100%

Awesome with Swift 2!

```
record.setObject(5, forKey: "numberOfClowns")
if let partyDate = record.objectForKey("date") as? NSDate {…}
```

```
record["numberOfClowns"] = 5
if let partyDate = record["date"] as? NSDate {…}
```

```
record["numberOfClowns"] = 5
if let partyDate = record["date"] as? NSDate {…}
```

```
modifyRecordsOperation.recordsToSave = ["I’m not a CKRecord!"]
```

```
record["numberOfClowns"] = 5
if let partyDate = record["date"] as? NSDate {…}
```

```
modifyRecordsOperation.recordsToSave = ["I’m not a CKRecord!"]
error: cannot assign a value of type '[String]' to a value of type '[CKRecord]?'
```

CloudKit

Storage architecture

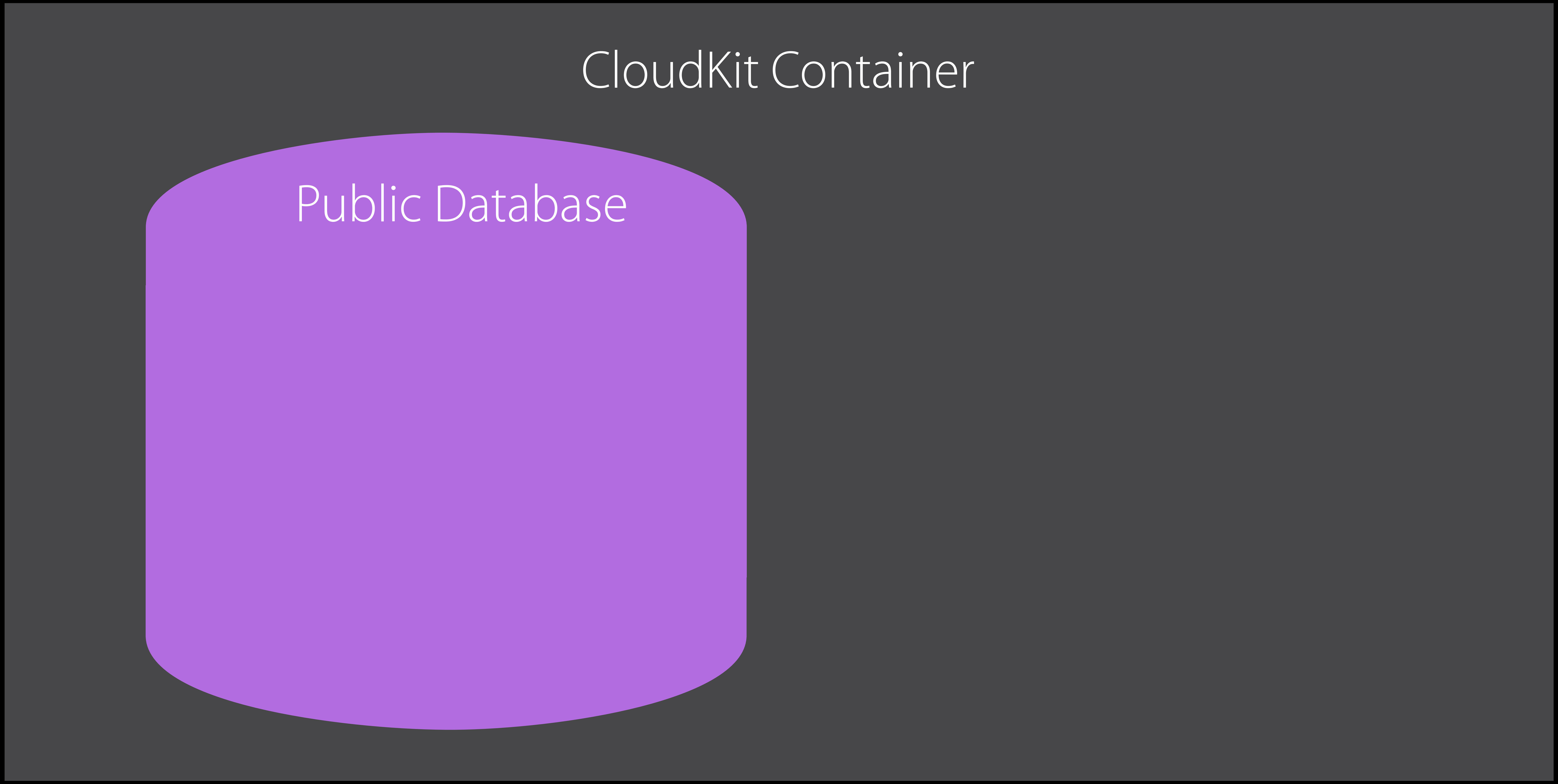
CloudKit

Storage architecture

CloudKit Container

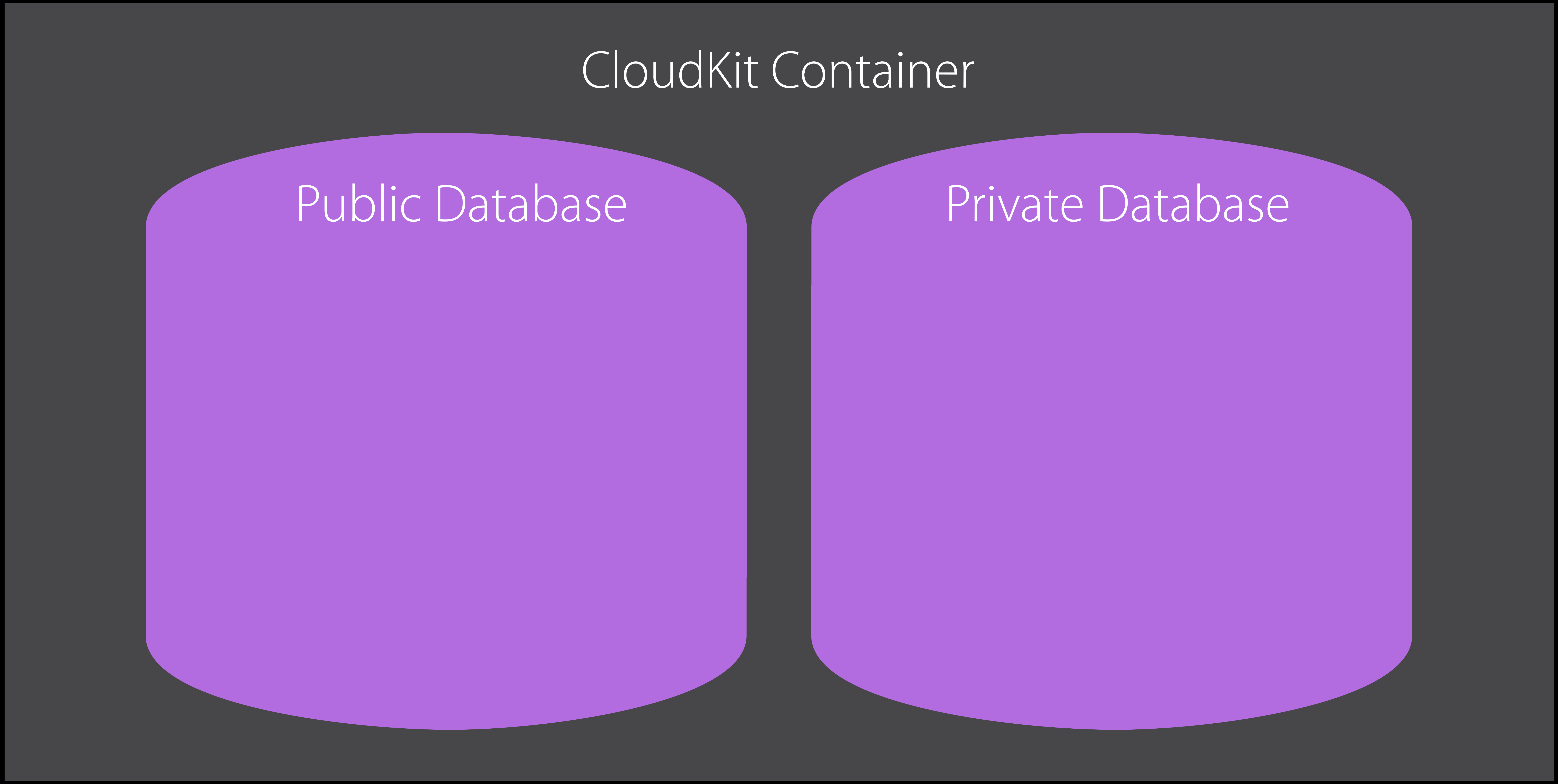
CloudKit

Storage architecture



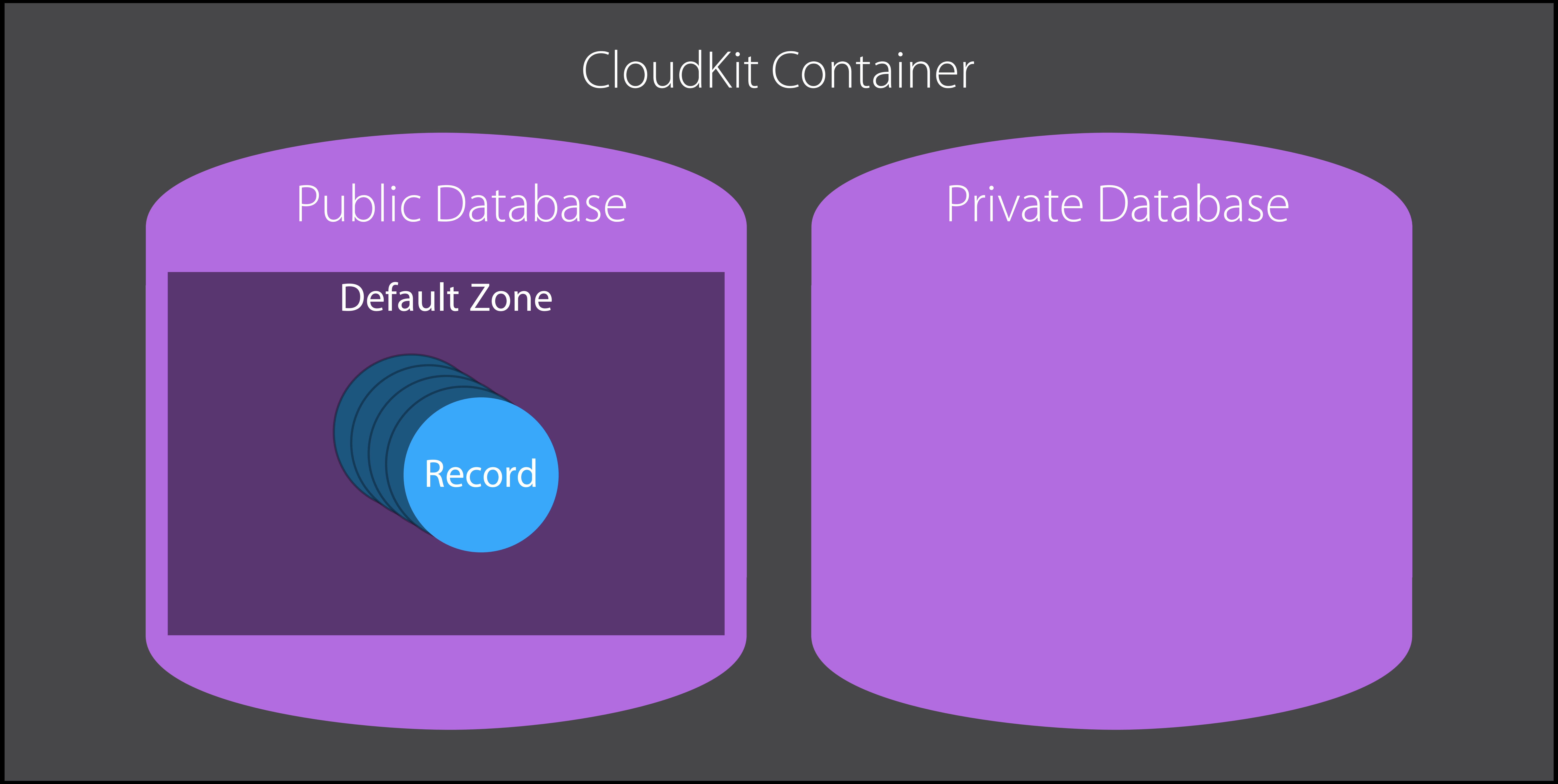
CloudKit

Storage architecture



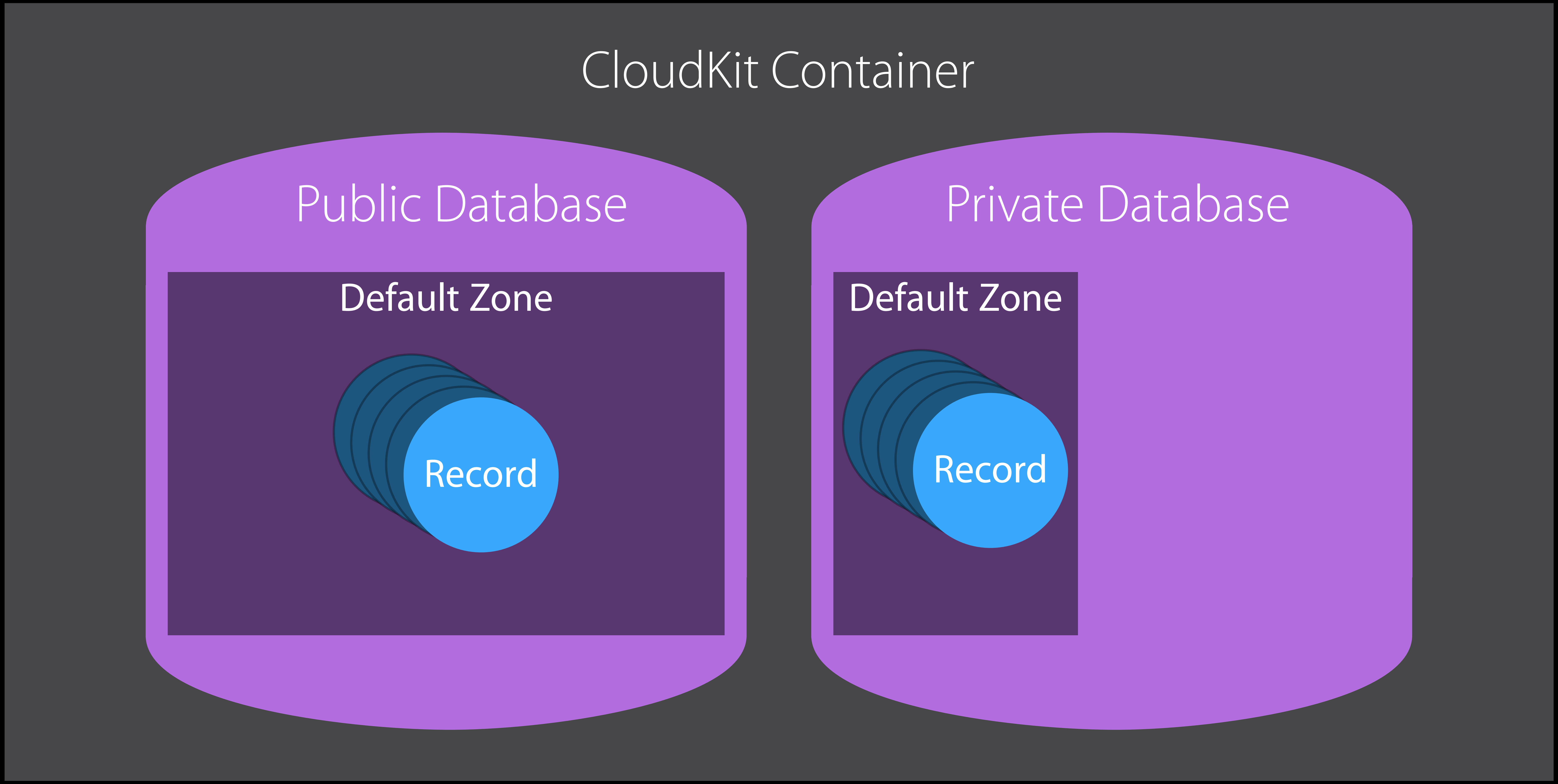
CloudKit

Storage architecture



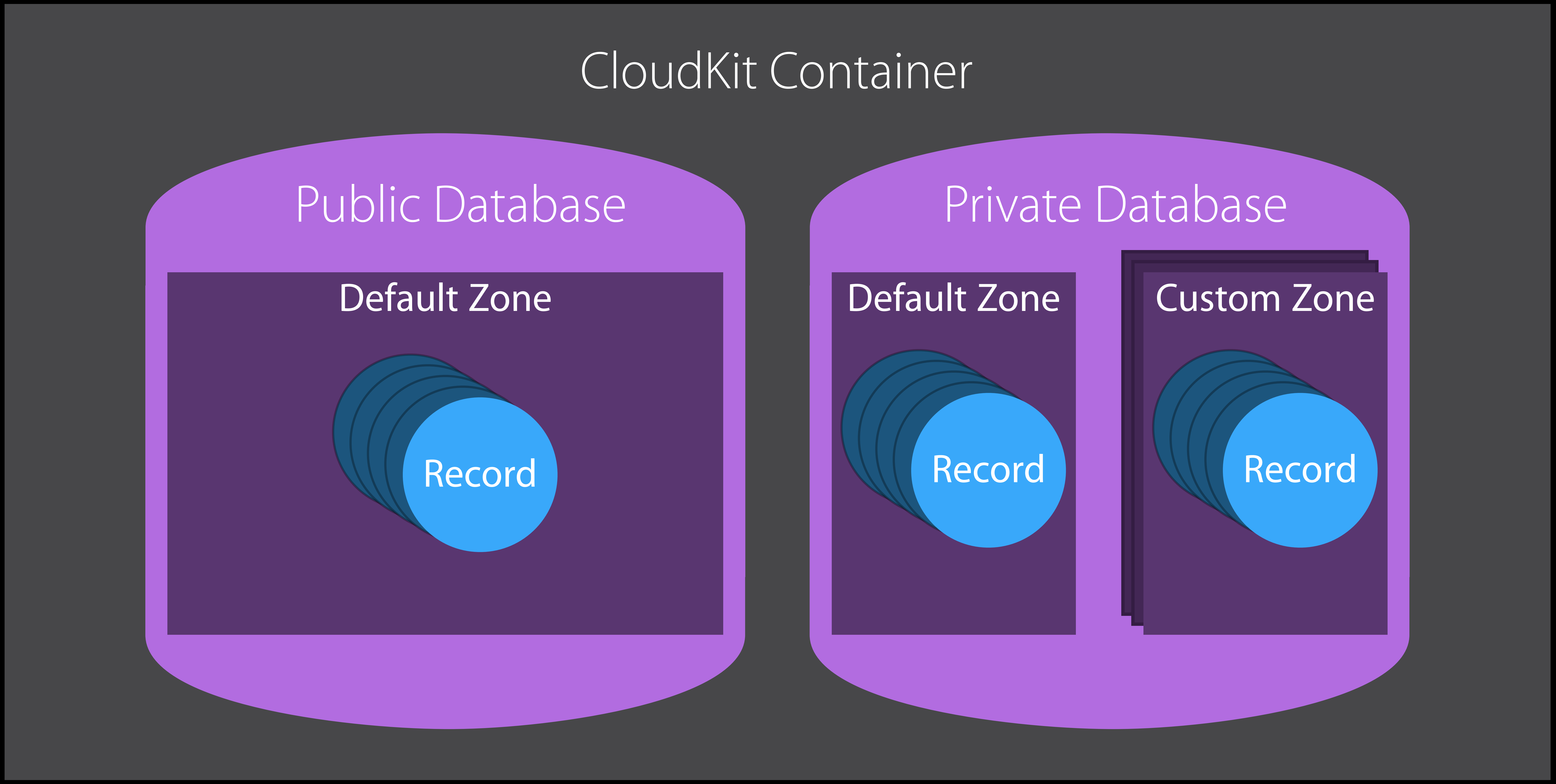
CloudKit

Storage architecture



CloudKit

Storage architecture

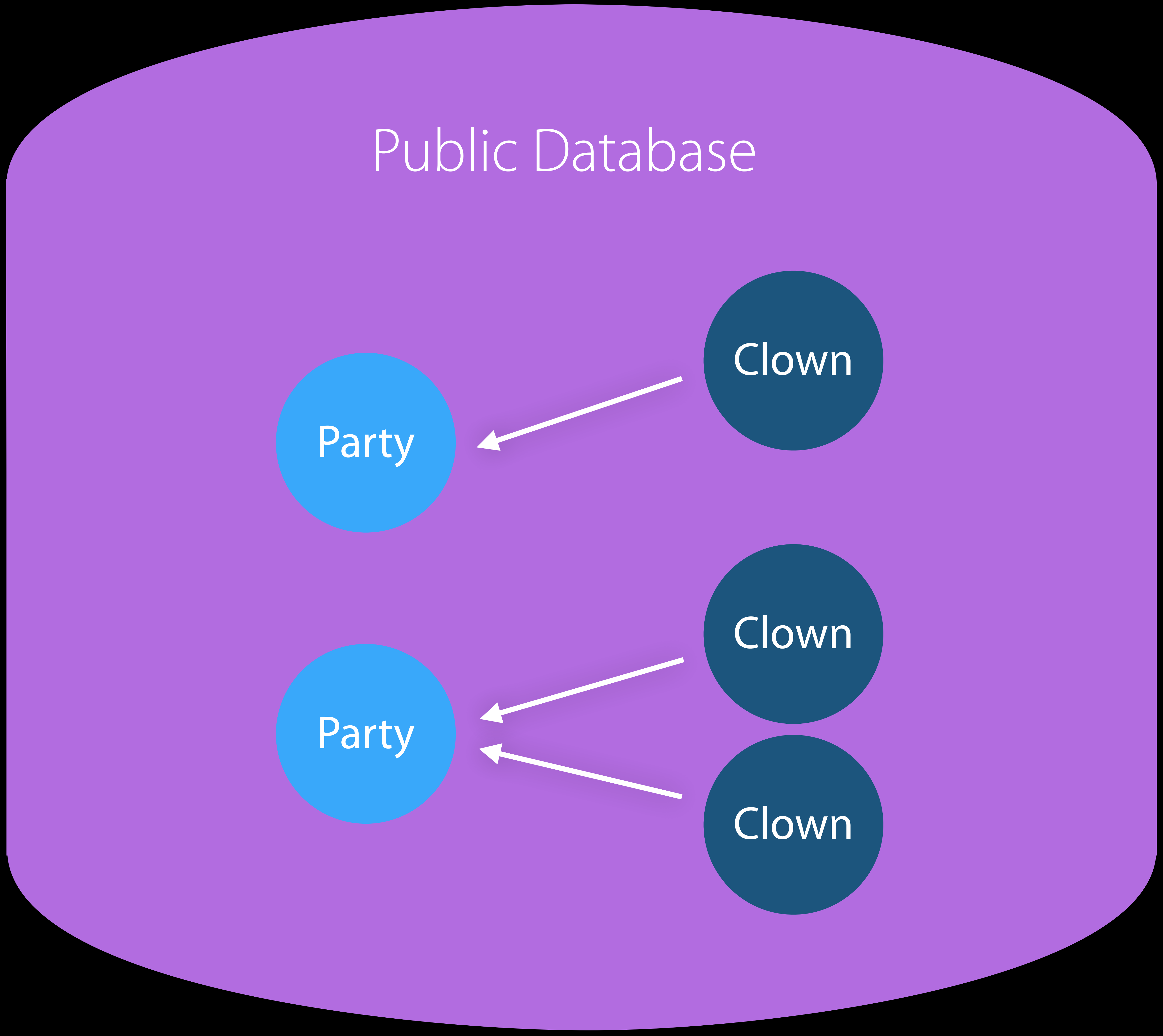


What We’ll Cover Today

Clown storage app

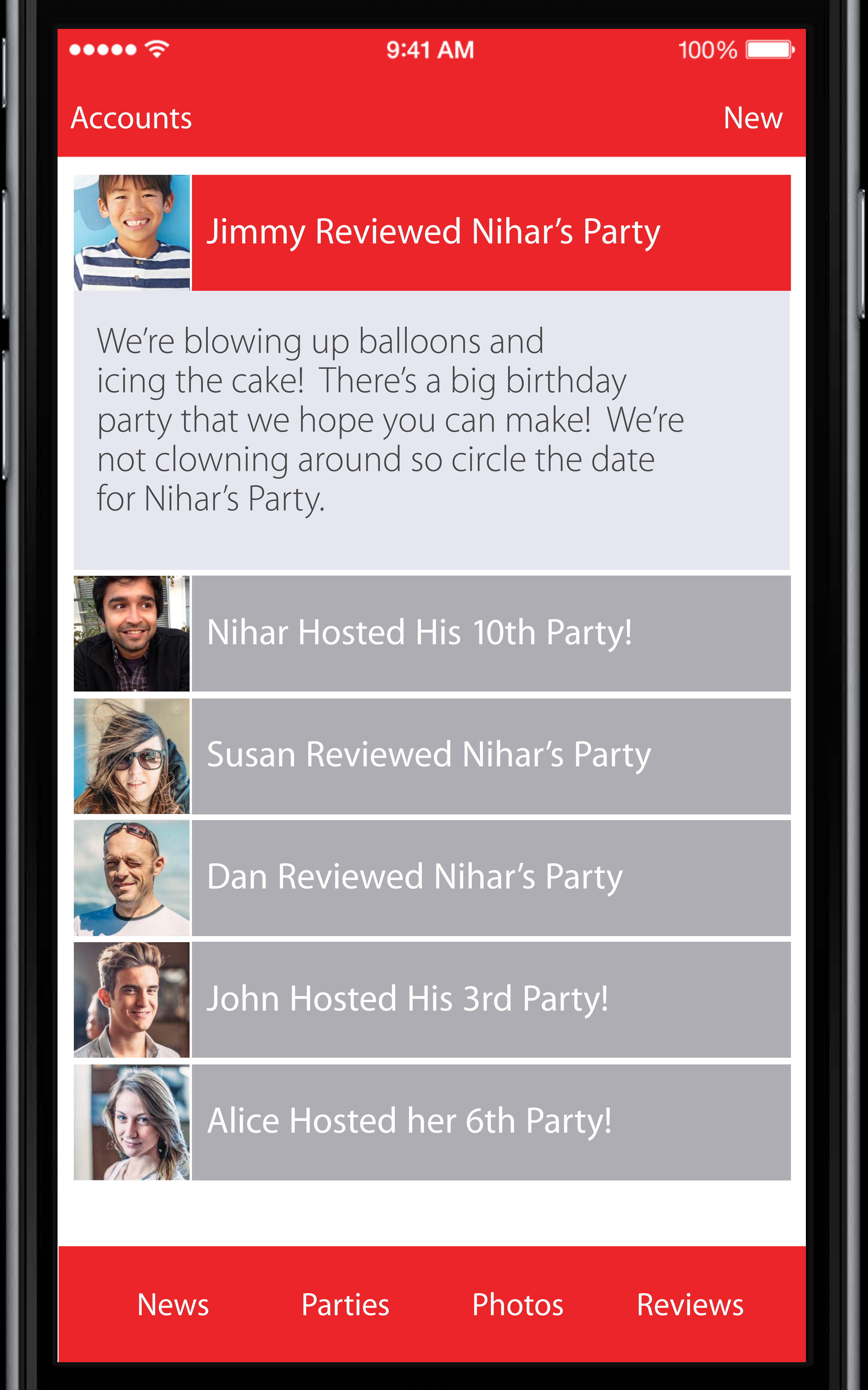
What We'll Cover Today

Clown storage app



ClownCentral

Our best fake clown app yet



Jimmy Reviewed Nihar’s Party

We’re blowing up balloons and icing the cake! There’s a big birthday party that we hope you can make! We’re not clowning around so circle the date for Nihar’s Party.

Nihar Hosted His 10th Party!

Susan Reviewed Nihar’s Party

Dan Reviewed Nihar’s Party

John Hosted His 3rd Party!

Alice Hosted her 6th Party!

What We'll Cover Today

CloudKit tips and tricks

What We'll Cover Today

CloudKit tips and tricks

Error handling

What We'll Cover Today

CloudKit tips and tricks

Error handling

Local cache

What We’ll Cover Today

CloudKit tips and tricks

Error handling

Local cache

Subscriptions

What We'll Cover Today

CloudKit tips and tricks

Error handling

Local cache

Subscriptions

Performance

Error Handling with CloudKit

As important as any feature!

Account Status

Account Status

ClownCentral requires iCloud account

Account Status

ClownCentral requires iCloud account

Private database access

Account Status

ClownCentral requires iCloud account

Private database access

Public database write access

Account Status

ClownCentral requires iCloud account

Private database access

Public database write access

```
container.accountStatusWithCompletionHandler { accountStatus, error in
…
}
```

Account Status

Account Status

```
CKErrorNotAuthenticated
```

Account Status

`CKErrorNotAuthenticated`

Check account status

Account Status

`CKErrorNotAuthenticated`

Check account status

- `CKAccountStatusNoAccount`

Account Status

NEW

`CKErrorNotAuthenticated`

Check account status

- `CKAccountStatusNoAccount`

`CKAccountChangedNotification`

Account Status

`CKErrorNotAuthenticated`

Check account status

- `CKAccountStatusNoAccount`

`CKAccountChangedNotification`

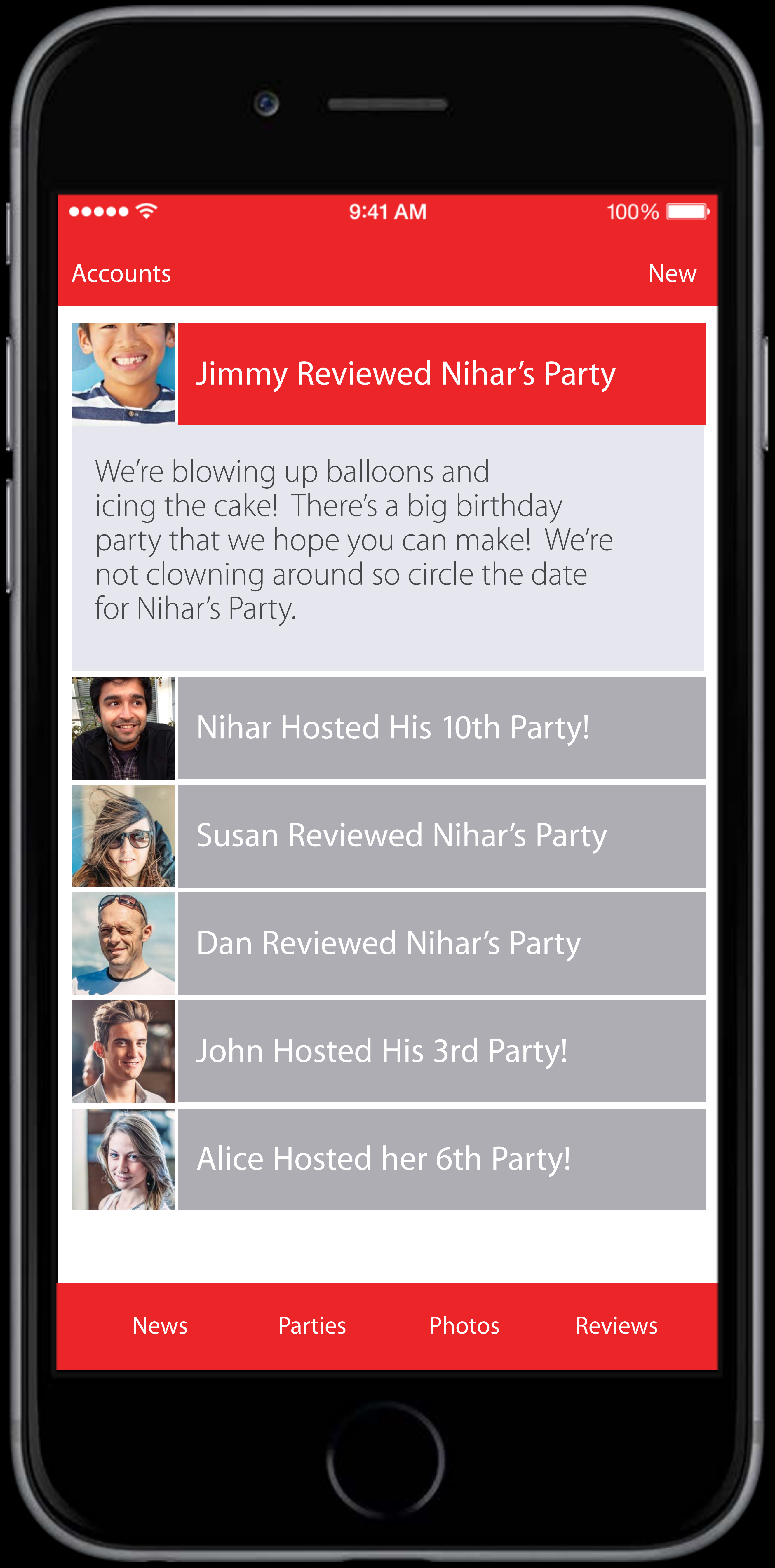
Account Status

`CKErrorNotAuthenticated`

Check account status

- `CKAccountStatusNoAccount`

`CKAccountChangedNotification`



Jimmy Reviewed Nihar's Party

We're blowing up balloons and icing the cake! There's a big birthday party that we hope you can make! We're not clowning around so circle the date for Nihar's Party.

Nihar Hosted His 10th Party!

Susan Reviewed Nihar's Party

Dan Reviewed Nihar's Party

John Hosted His 3rd Party!

Alice Hosted her 6th Party!

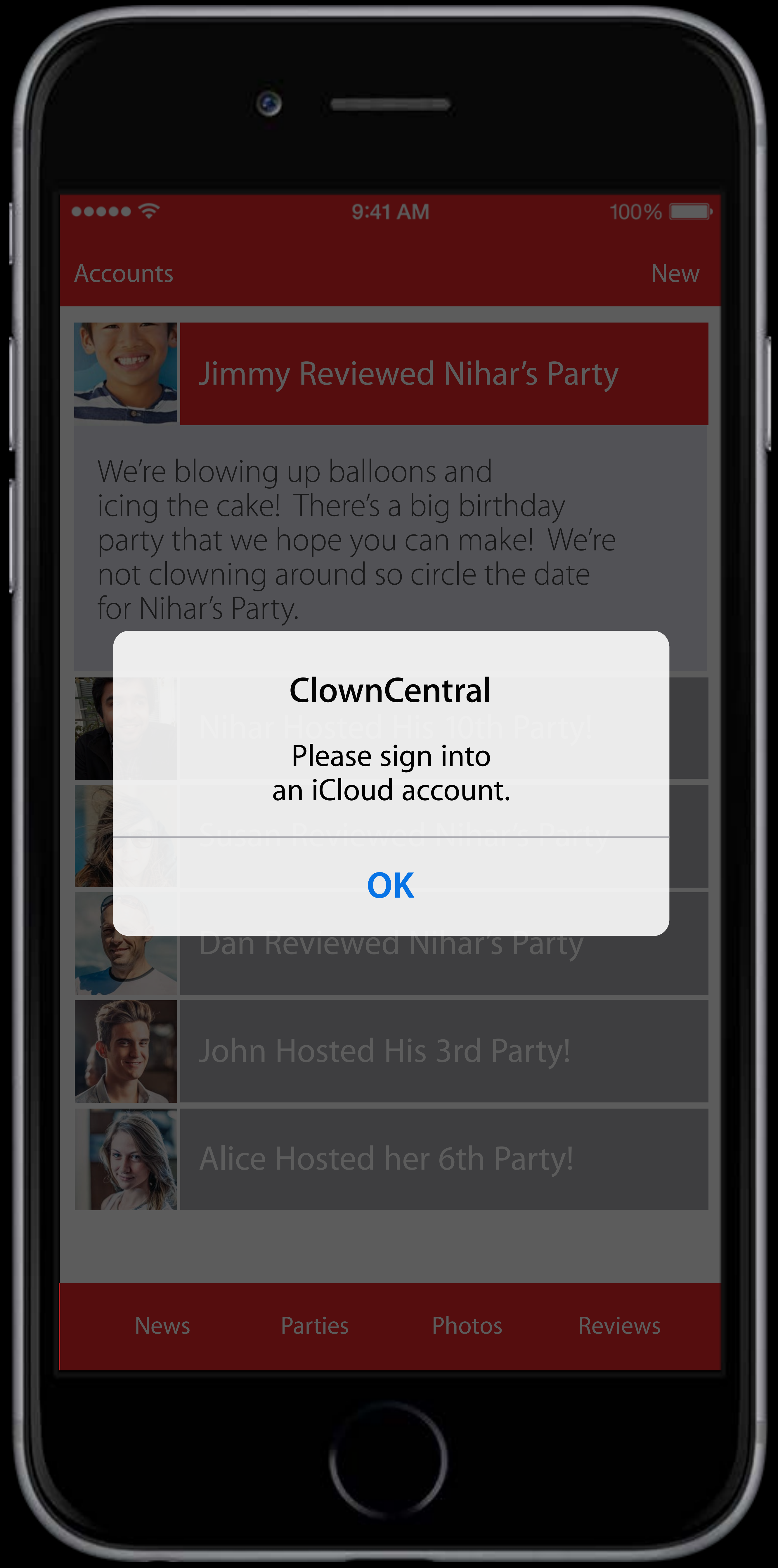
Account Status

`CKErrorNotAuthenticated`

Check account status

- `CKAccountStatusNoAccount`

`CKAccountChangedNotification`



Jimmy Reviewed Nihar's Party

We're blowing up balloons and icing the cake! There's a big birthday party that we hope you can make! We're not clowning around so circle the date for Nihar's Party.

Dan Reviewed Nihar's Party

John Hosted His 3rd Party!

Alice Hosted her 6th Party!

ClownCentral

Please sign into
an iCloud account.

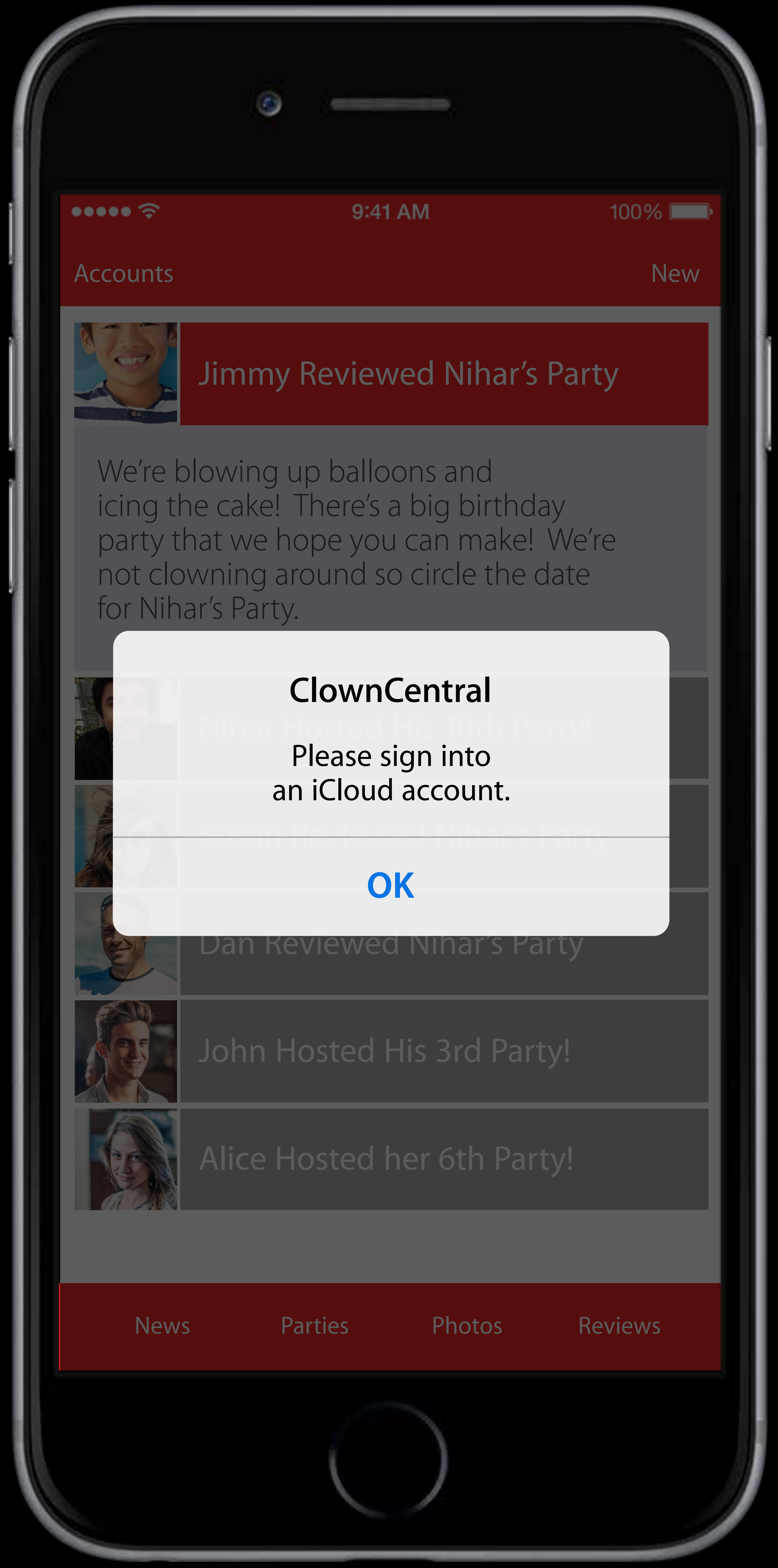
Account Status

`CKErrorNotAuthenticated`

Check account status

- `CKAccountStatusNoAccount`

`CKAccountChangedNotification`



We're blowing up balloons and icing the cake! There's a big birthday party that we hope you can make! We're not clowning around so circle the date for Nihar's Party.

ClownCentral

Please sign into an iCloud account.

OK

9:41 AM
100%
Accounts
New
Jimmy Reviewed Nihar's Party
Dan Reviewed Nihar's Party
John Hosted His 3rd Party!
Alice Hosted her 6th Party!
News
Parties
Photos
Reviews

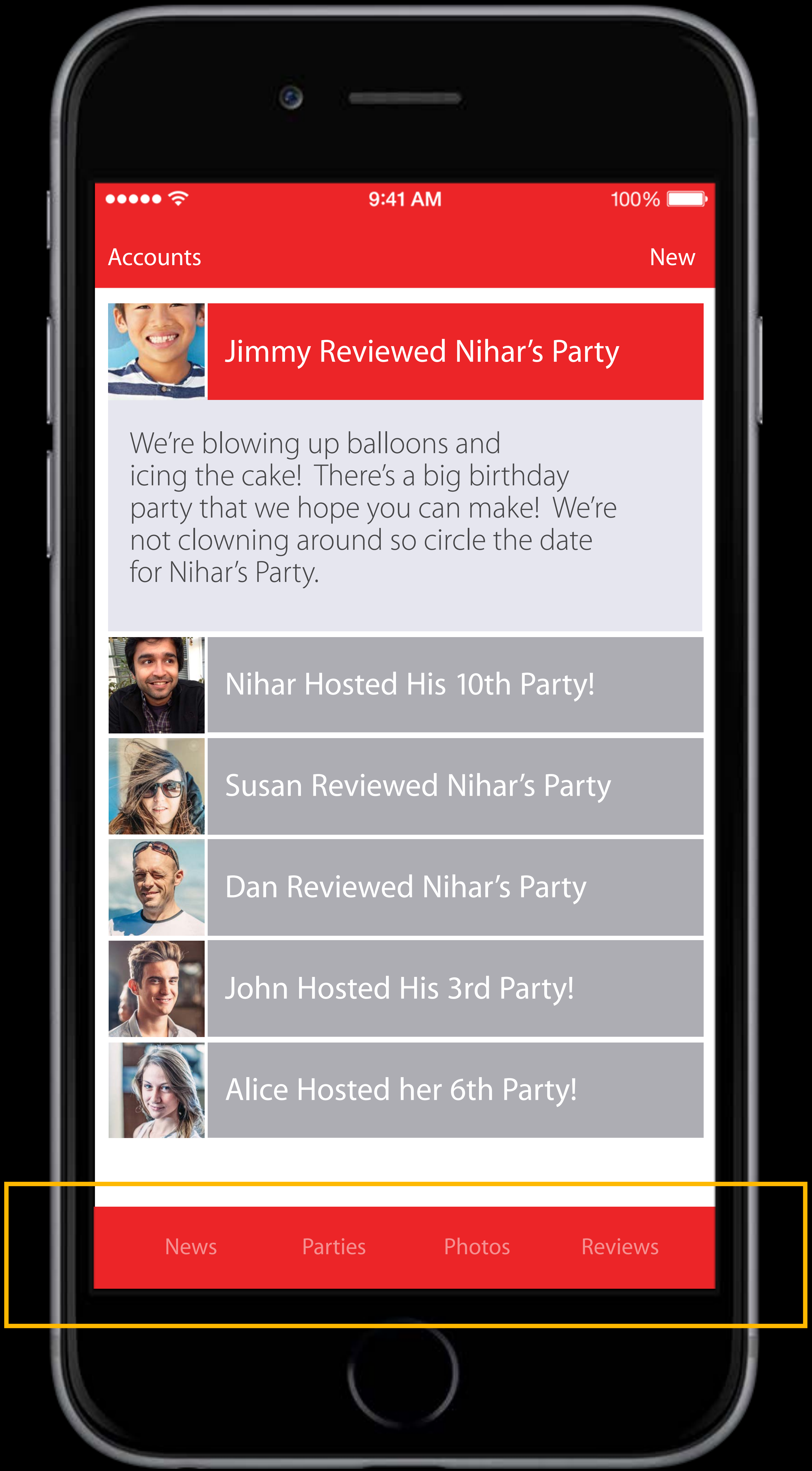
Account Status

`CKErrorNotAuthenticated`

Check account status

- `CKAccountStatusNoAccount`

`CKAccountChangedNotification`



Jimmy Reviewed Nihar’s Party

We’re blowing up balloons and icing the cake! There’s a big birthday party that we hope you can make! We’re not clowning around so circle the date for Nihar’s Party.

Nihar Hosted His 10th Party!

Susan Reviewed Nihar’s Party

Dan Reviewed Nihar’s Party

John Hosted His 3rd Party!

Alice Hosted her 6th Party!

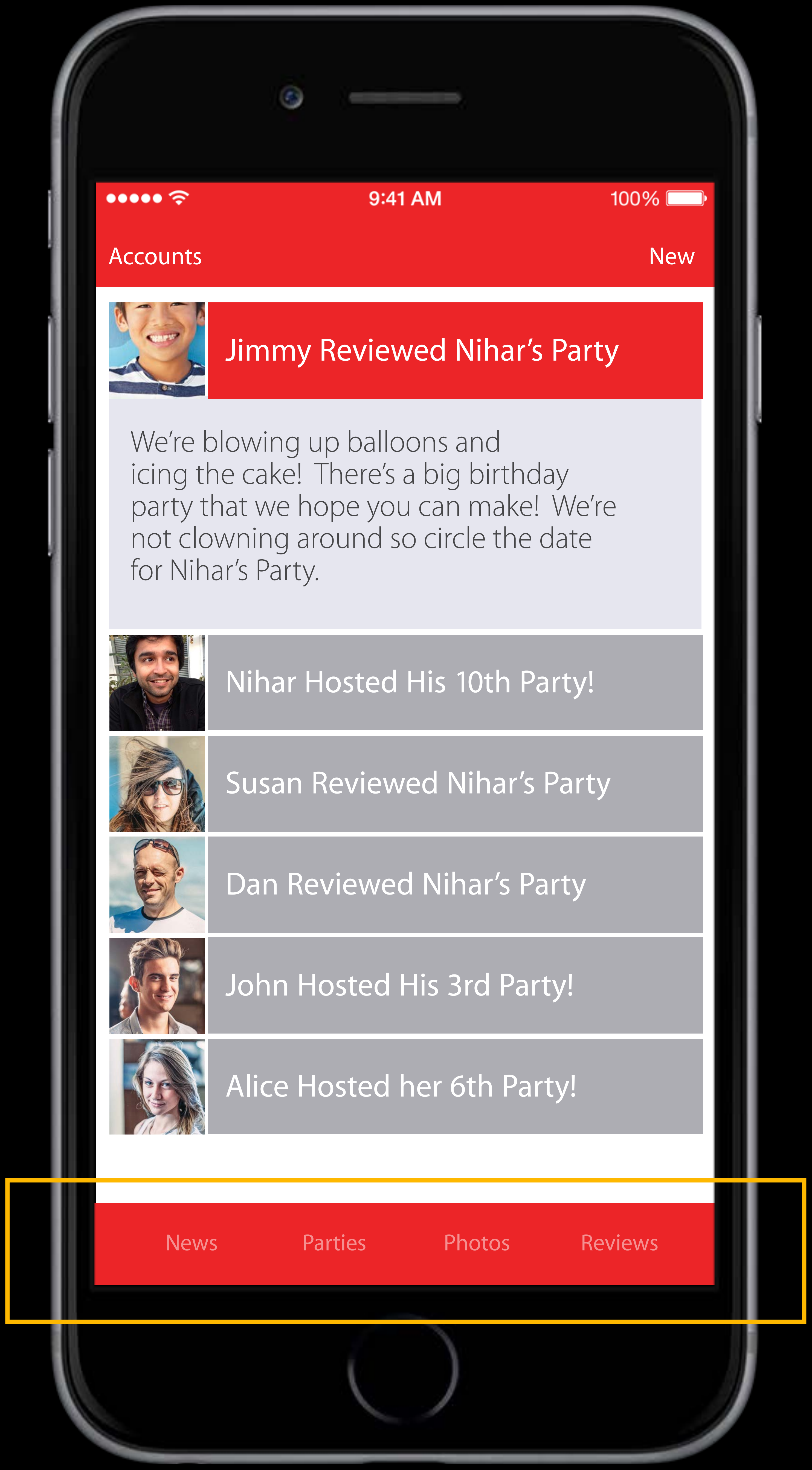
Account Status

`CKErrorNotAuthenticated`

Check account status

- `CKAccountStatusNoAccount`

`CKAccountChangedNotification`


●●●●● 9:41 AM 100%
Accounts
New
Jimmy Reviewed Nihar's Party
Nihar Hosted His 10th Party!
Susan Reviewed Nihar's Party
Dan Reviewed Nihar's Party
John Hosted His 3rd Party!
Alice Hosted her 6th Party!
News
Parties
Photos
Reviews

We're blowing up balloons and icing the cake! There's a big birthday party that we hope you can make! We're not clowning around so circle the date for Nihar's Party.

✓

Retrying Operations

Retrying Operations

Poor network conditions

- `CKErrorNetworkFailure`

Retrying Operations

Poor network conditions

- `CKErrorNetworkFailure`

Busy servers

- `CKErrorServiceUnavailable`
- `CKErrorZoneBusy`

Retrying Operations

Retrying Operations

CKErrorRetryAfterKey

```
if let retryAfterValue = error.userInfo[CKErrorRetryAfterKey] as? Double {
    let retryAfterDate = NSDate(timeIntervalSinceNow: retryAfterValue)
}
```

Rate Limiting

Rate Limiting

`CKErrorRequestRateLimited`

Rate Limiting

`CKErrorRequestRateLimited`

Mitigates application bugs

Rate Limiting

`CKErrorRequestRateLimited`

Mitigates application bugs

Client-side throttle

Rate Limiting

`CKErrorRequestRateLimited`

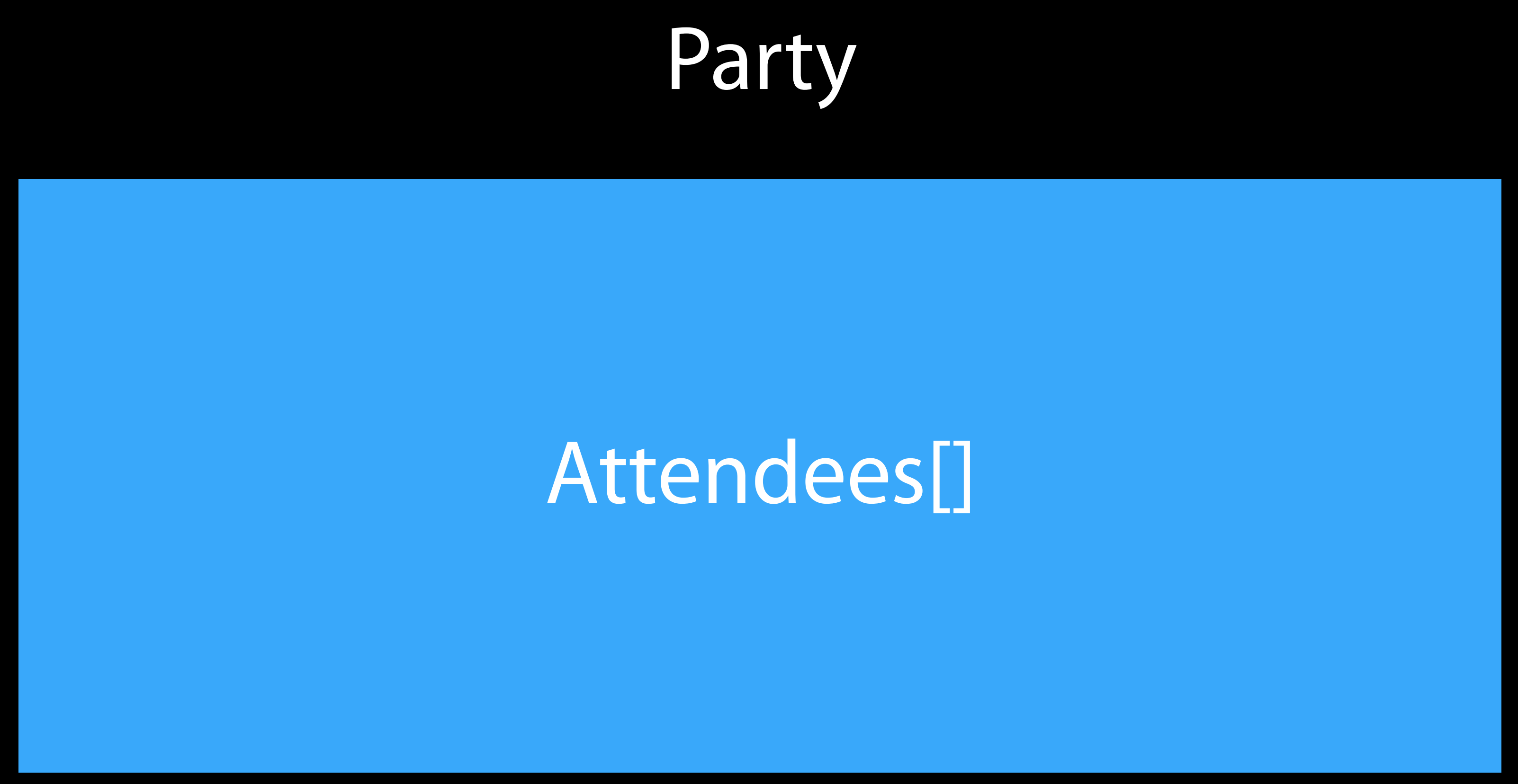
Mitigates application bugs

Client-side throttle

`CKErrorRetryAfterKey`

Handling Conflicts

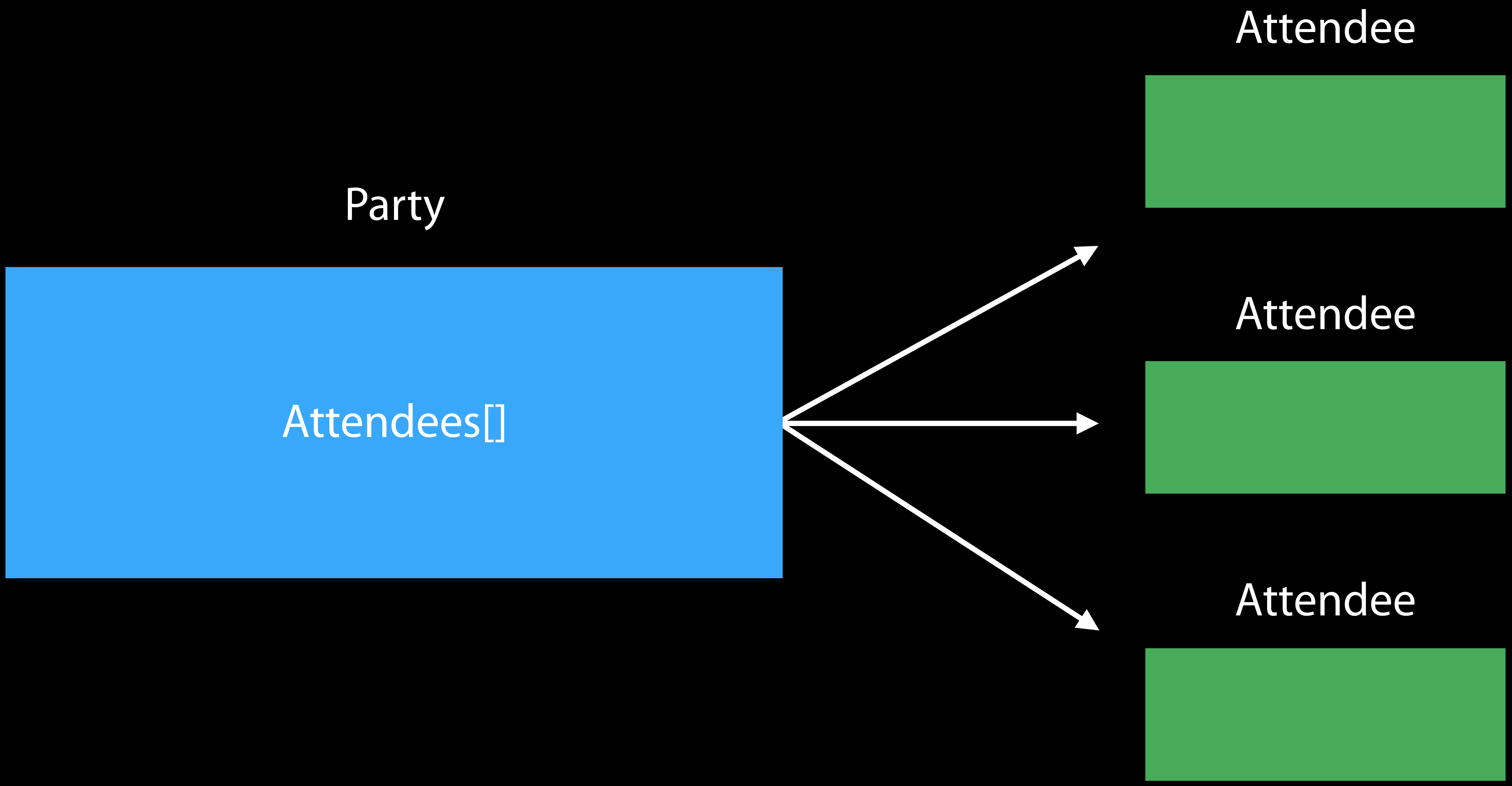
Handling Conflicts



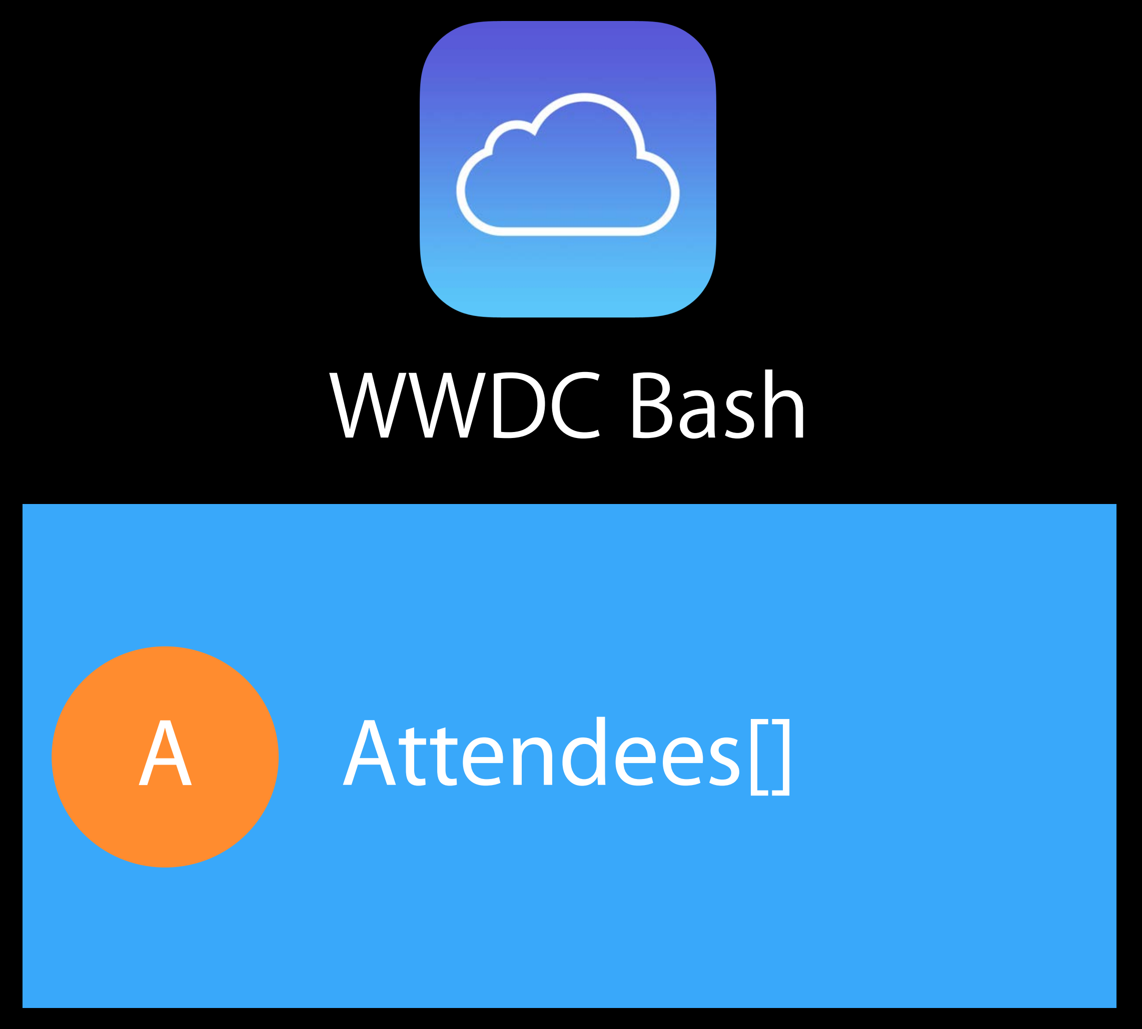
Party

Attendees[]

Handling Conflicts



Handling Conflicts

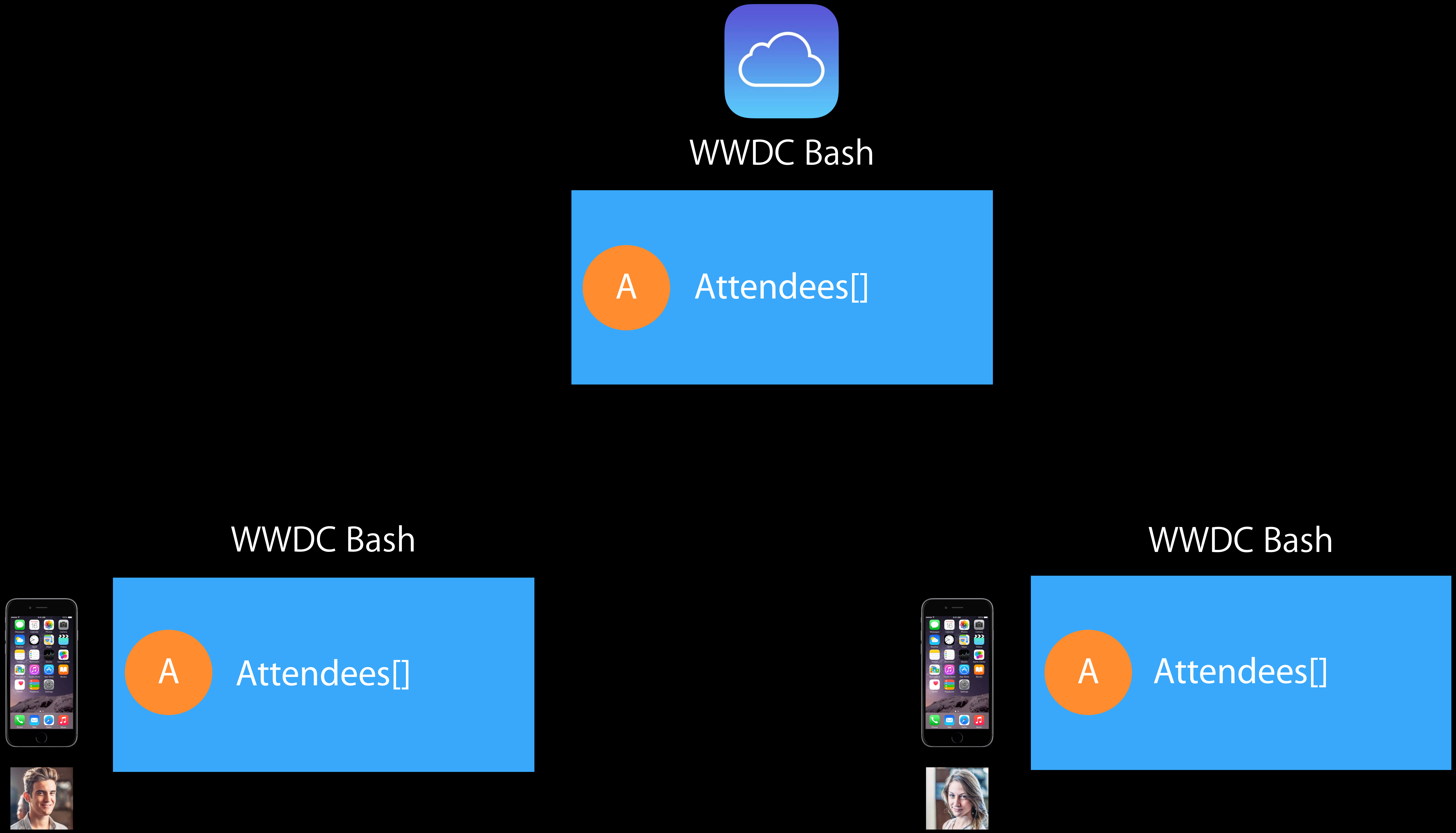


WWDC Bash

A

Attendees[]

Handling Conflicts



WWDC Bash

A Attendees[]

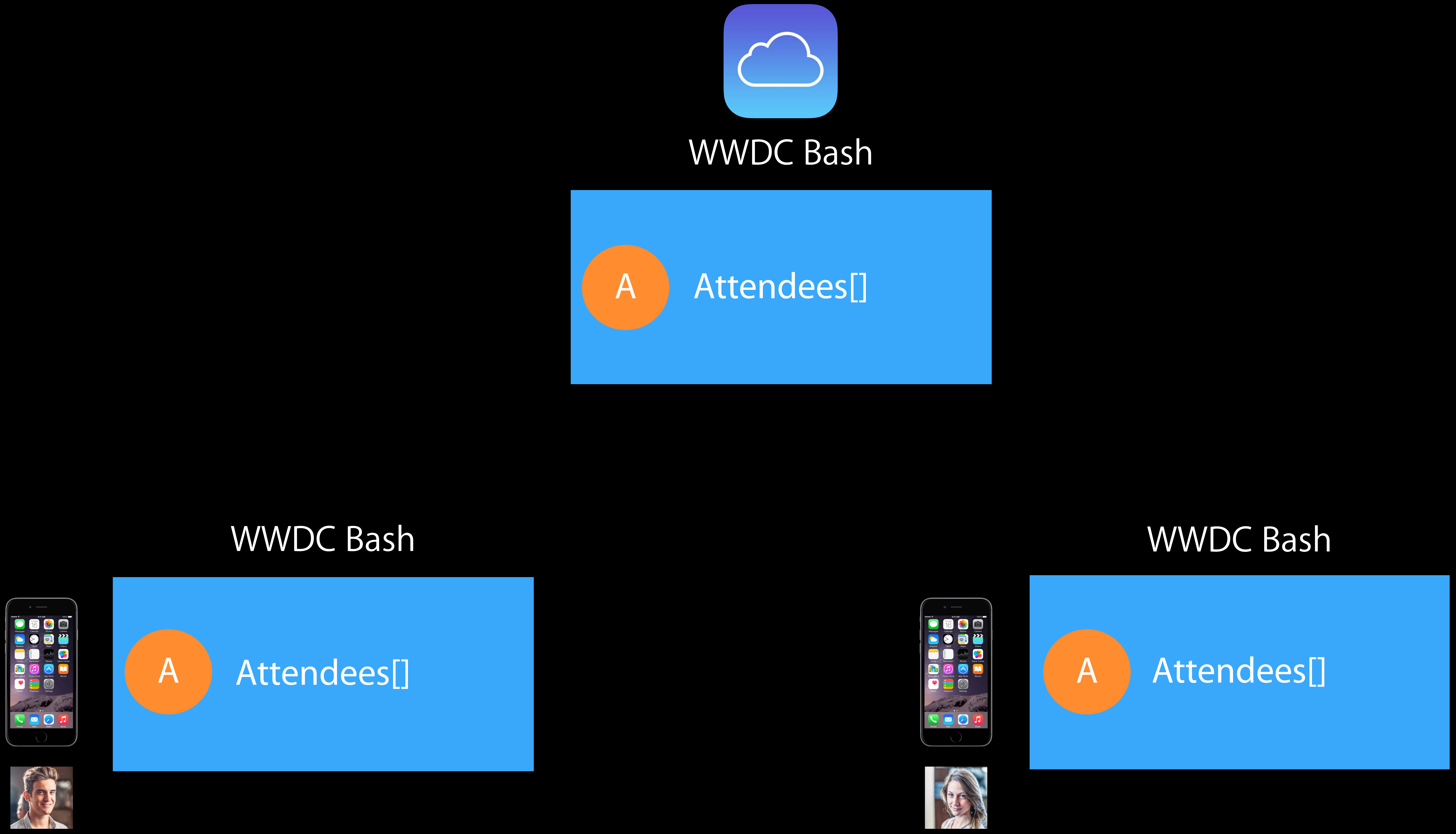
WWDC Bash

A Attendees[]

WWDC Bash

A Attendees[]

Handling Conflicts



WWDC Bash

A Attendees[]

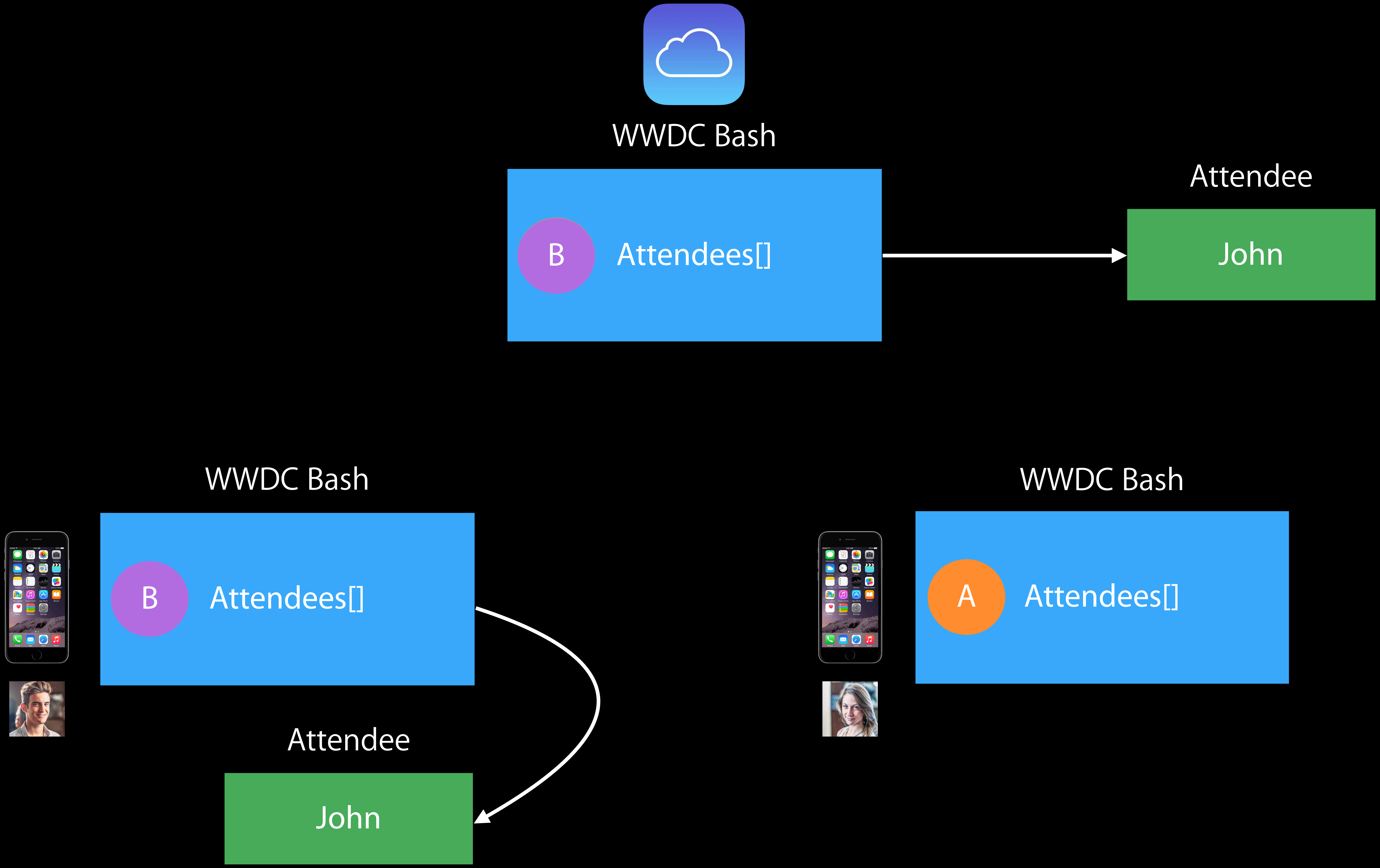
WWDC Bash

A Attendees[]

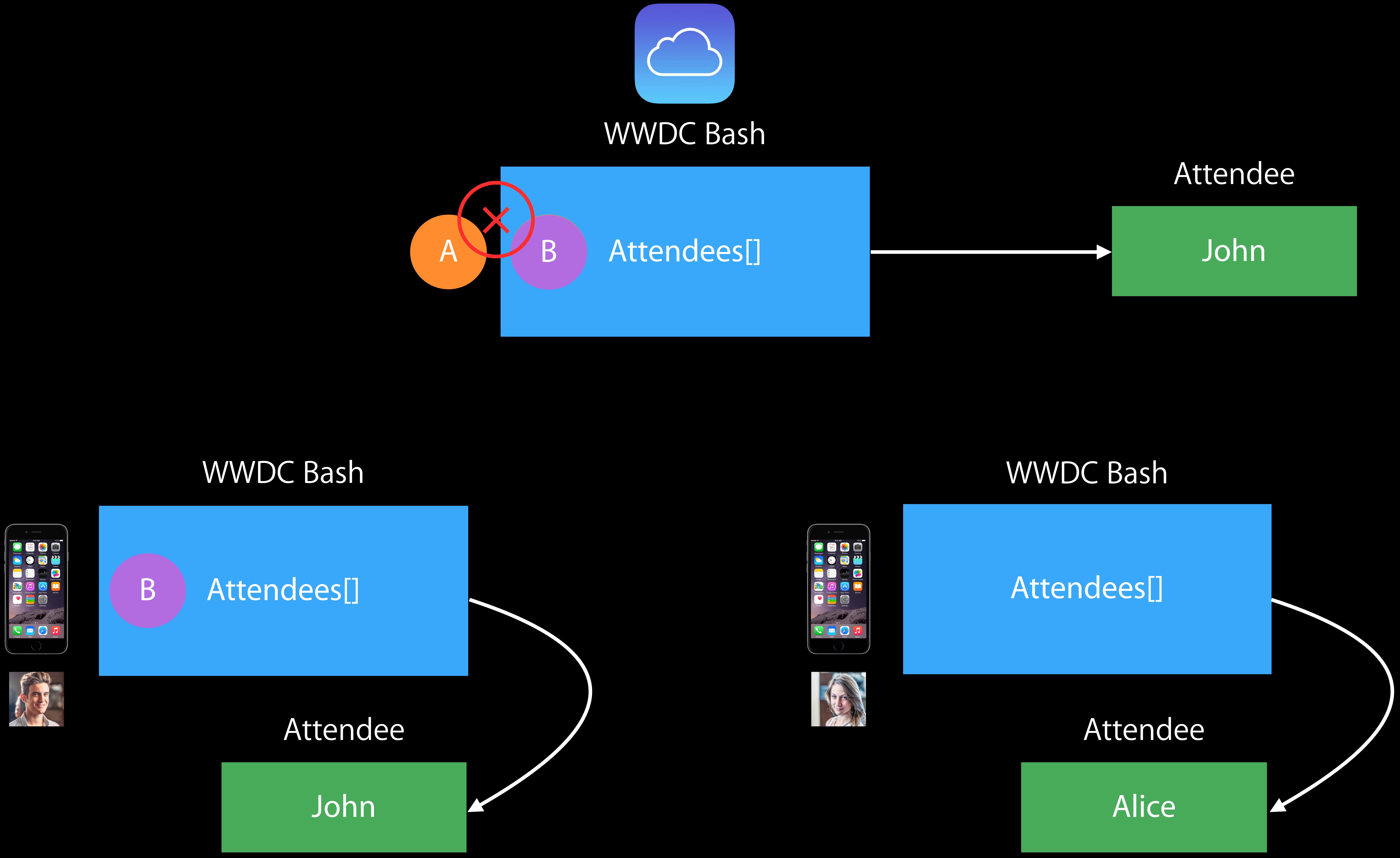
WWDC Bash

A Attendees[]

Handling Conflicts



Handling Conflicts



Handling Conflicts

Handling Conflicts

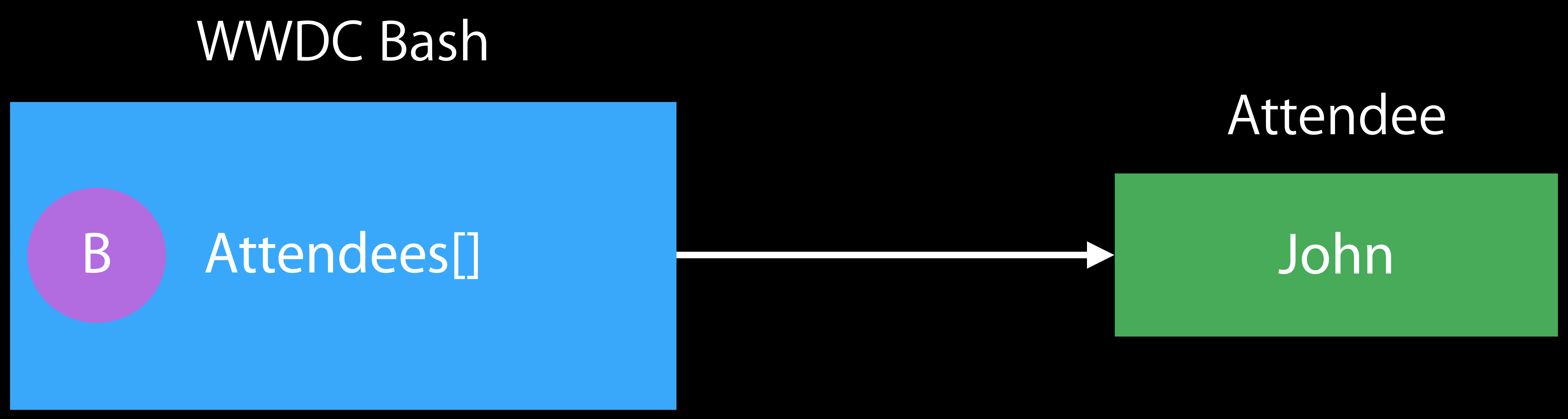
`CKErrorServerRecordChanged`

Handling Conflicts

```
CKErrorServerRecordChanged

if let serverRecord =
error.userInfo[CKRecordChangedErrorServerRecordKey] as? CKRecord {







}
```

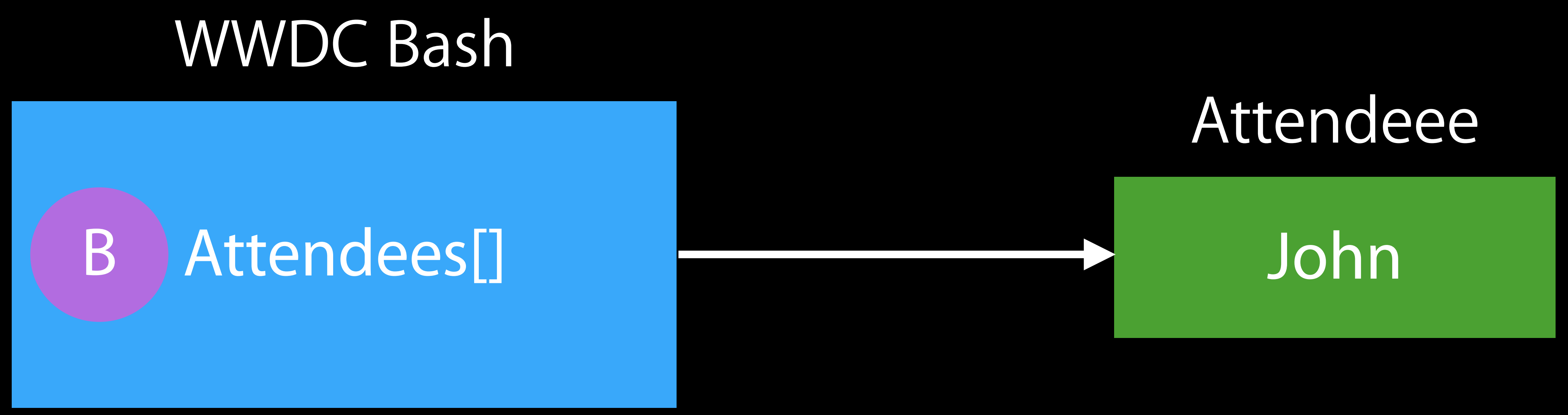
Handling Conflicts

```
CKErrorServerRecordChanged

if let serverRecord =
error.userInfo[CKRecordChangedErrorServerRecordKey] as? CKRecord {
```

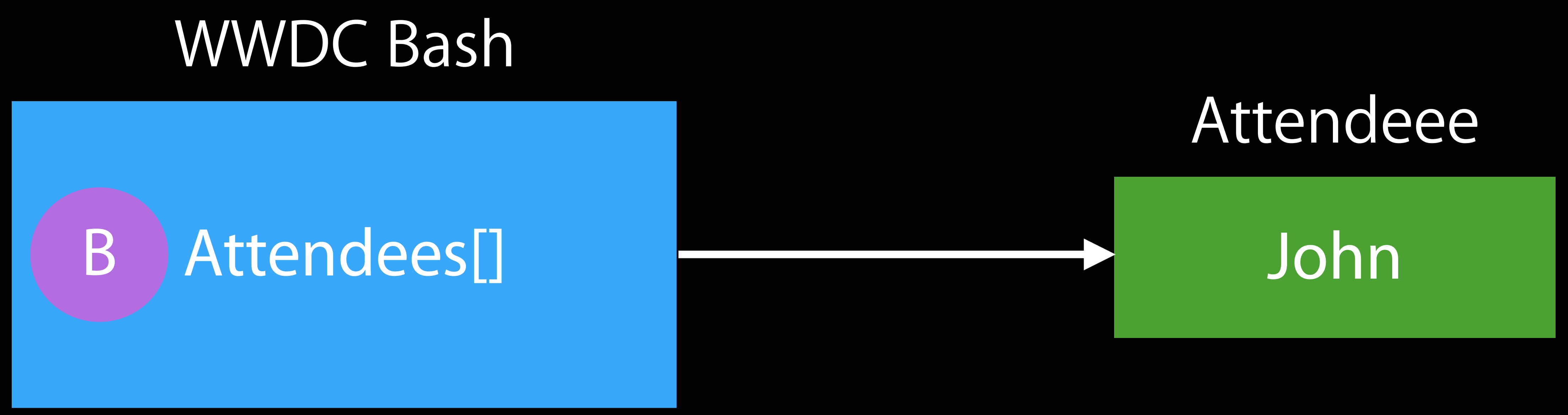


```
}
```

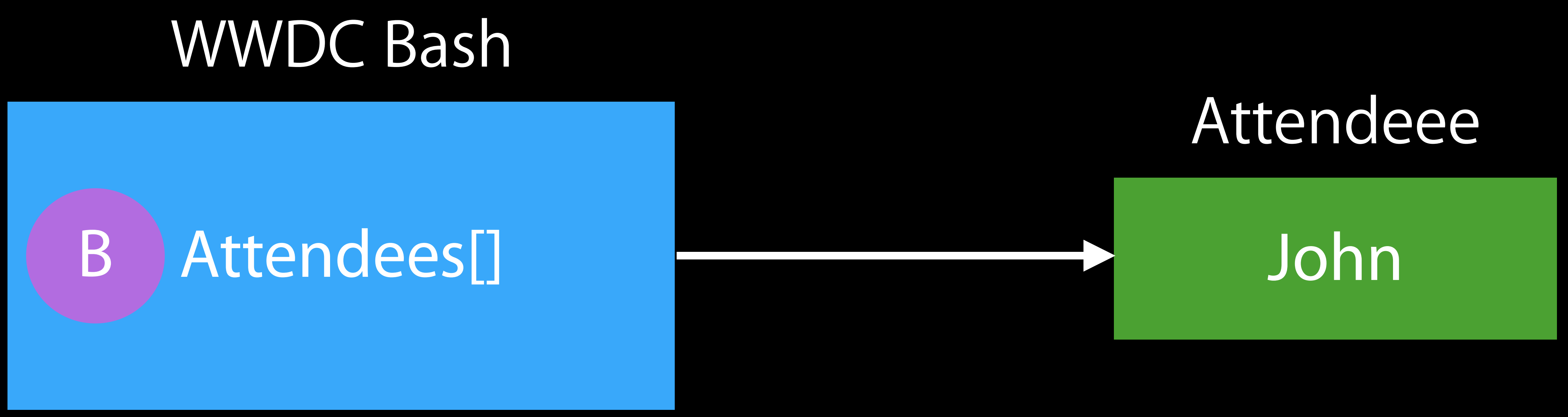
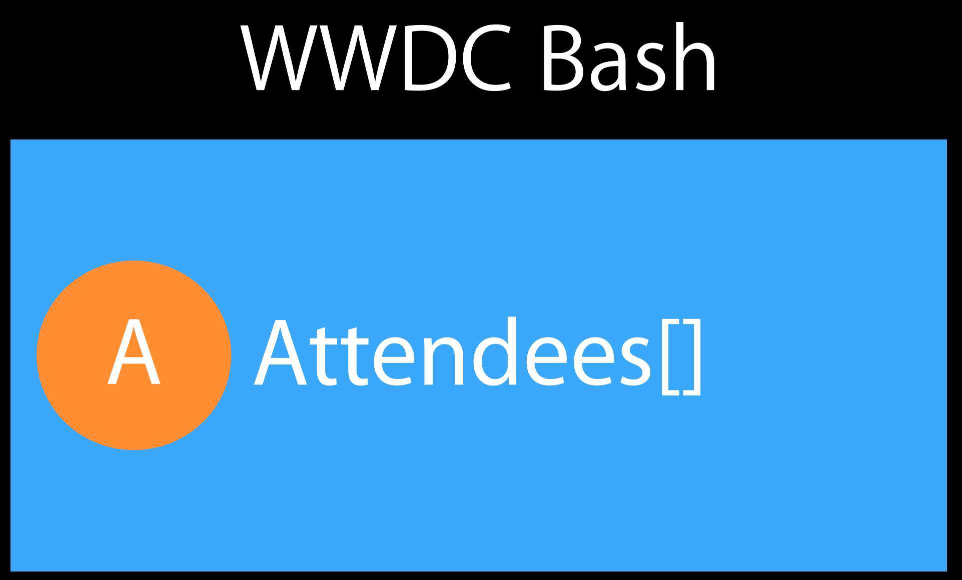
Handling Conflicts

Key	Record
`ServerRecordKey`	 WWDC Bash B Attendees[] Attendeee John

Handling Conflicts

Key	Record
`ServerRecordKey`	
`AncestorRecordKey`	

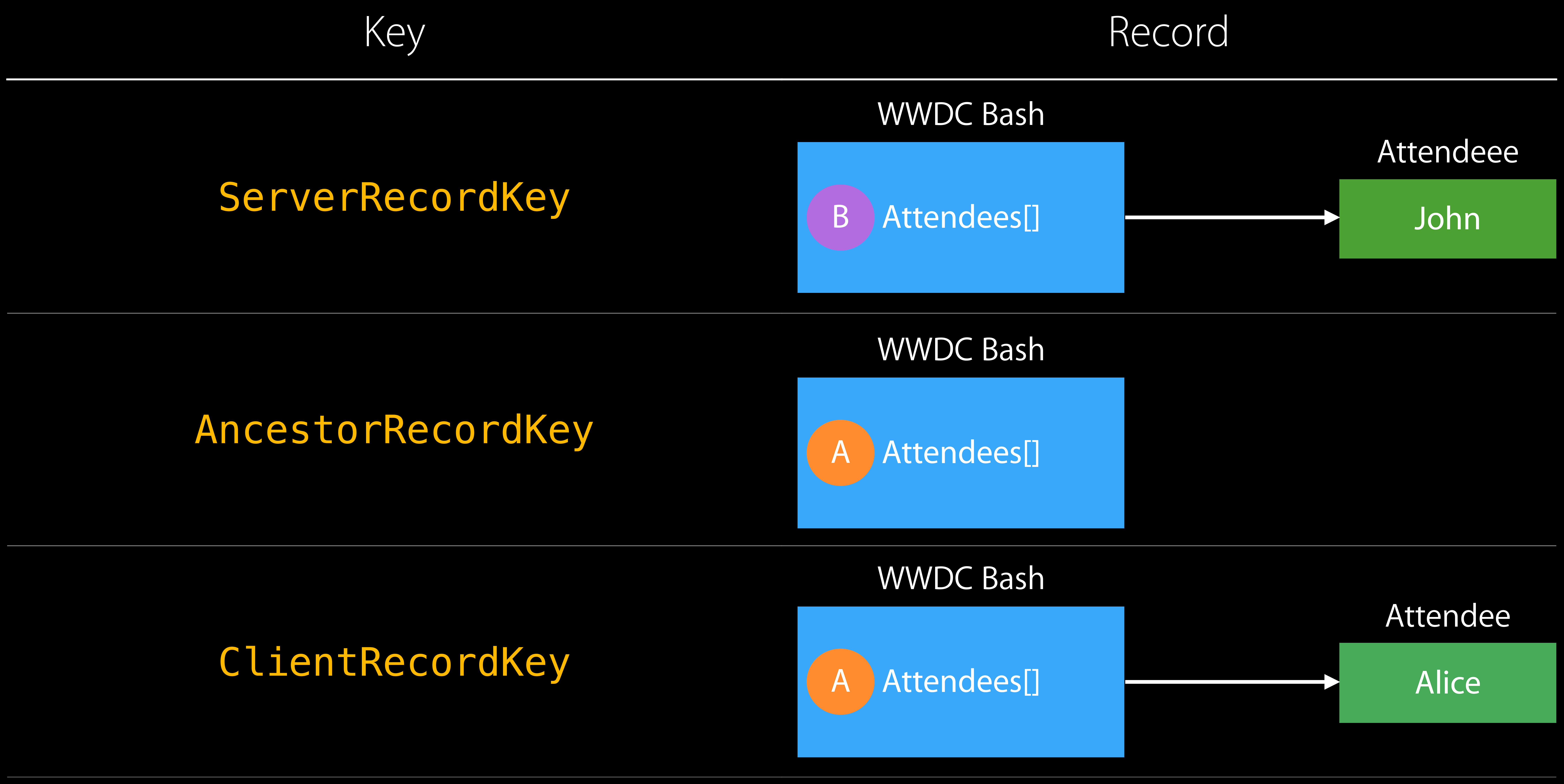
Handling Conflicts

Key	Record
ServerRecordKey	 WWDC Bash B Attendees[] Attendeee John
AncestorRecordKey	 WWDC Bash A Attendees[]

Handling Conflicts

Key	Record
`ServerRecordKey`	WWDC Bash B Attendees[] → Attendeee John
`AncestorRecordKey`	WWDC Bash A Attendees[]
`ClientRecordKey`	

Handling Conflicts



Key

Record

ServerRecordKey

WWDC Bash

B Attendees[]

Attendeee

John

AncestorRecordKey

WWDC Bash

A Attendees[]

ClientRecordKey

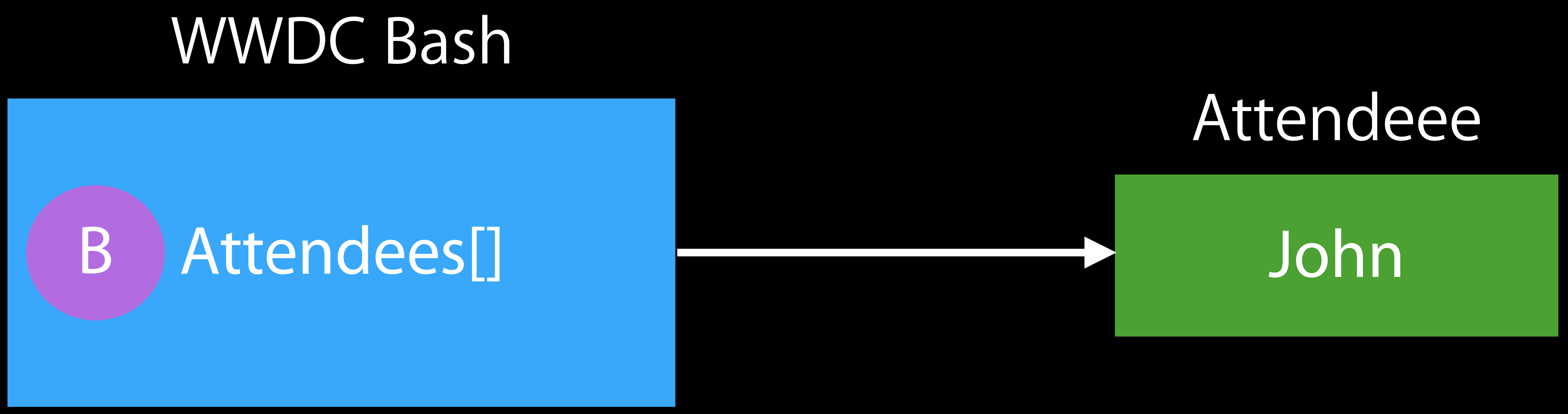
WWDC Bash

A Attendees[]

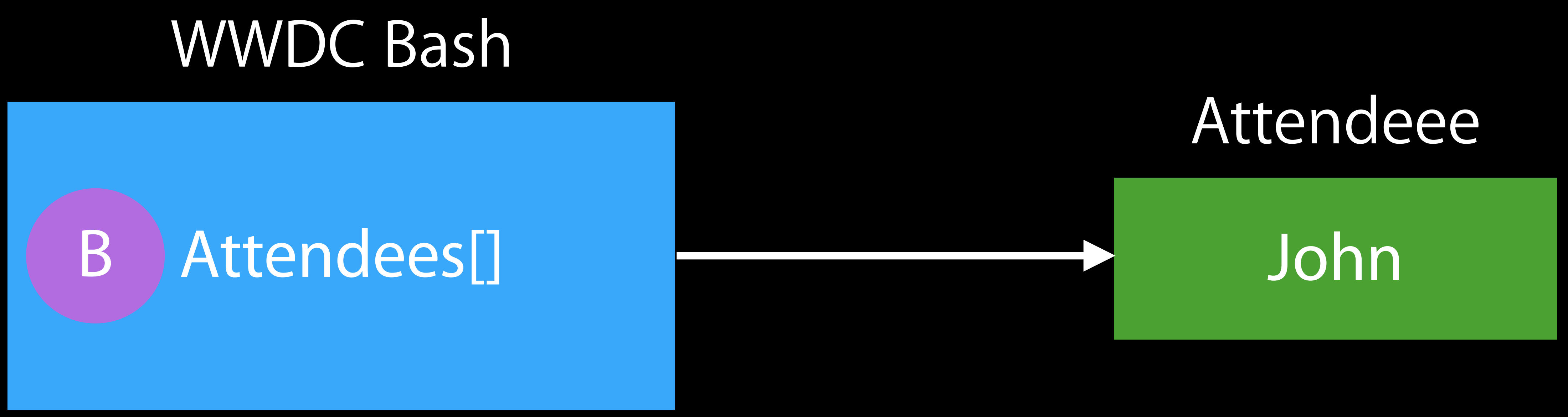
Attendee

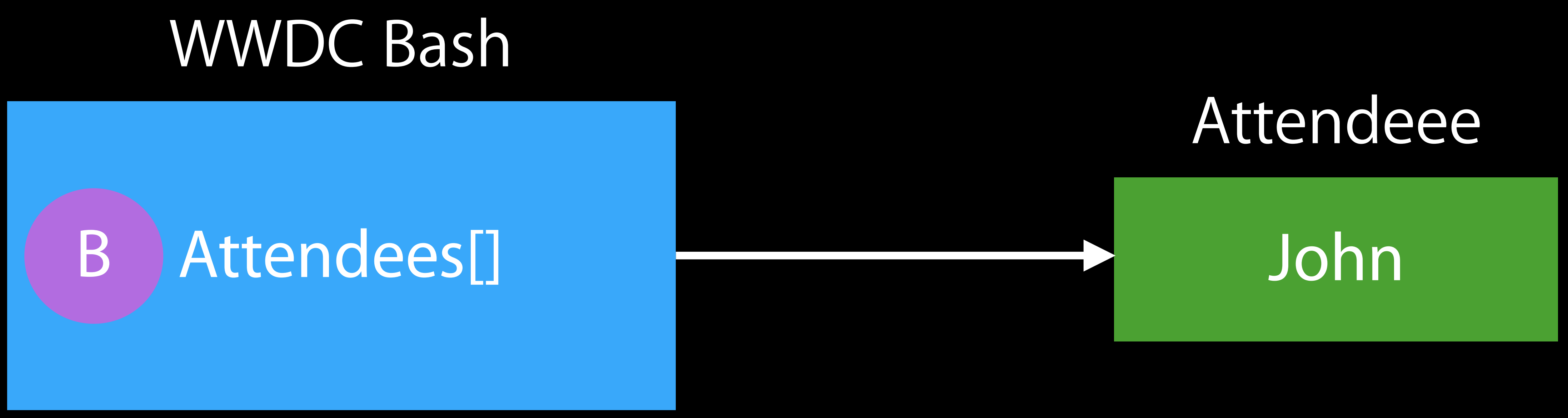
Alice

Handling Conflicts

Key	Record
`ServerRecordKey`	

Handling Conflicts

Key	Record
`ServerRecordKey`	 WWDC Bash B Attendees[] Attendeee John



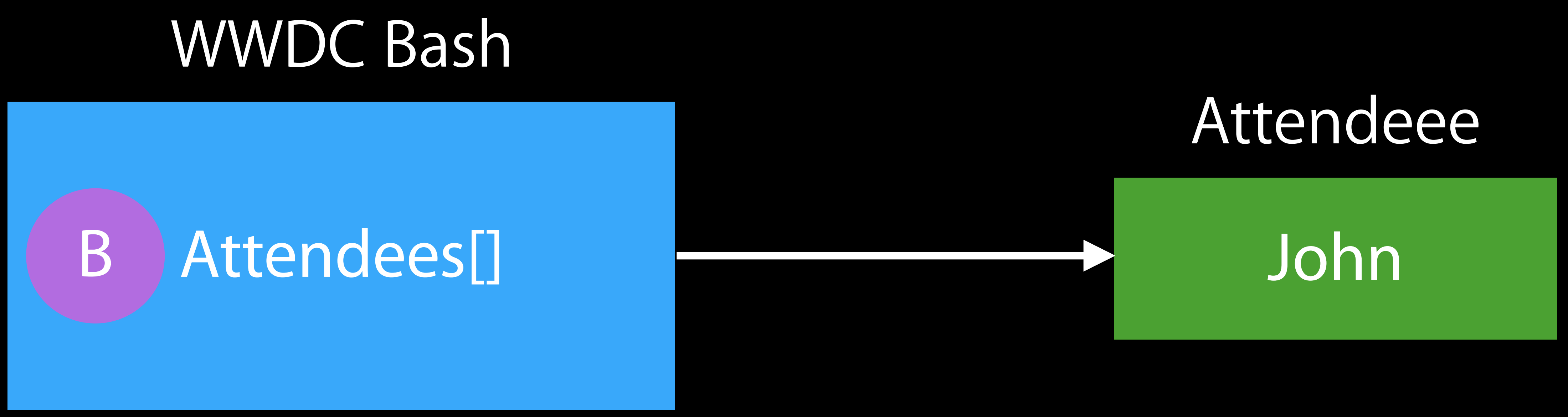
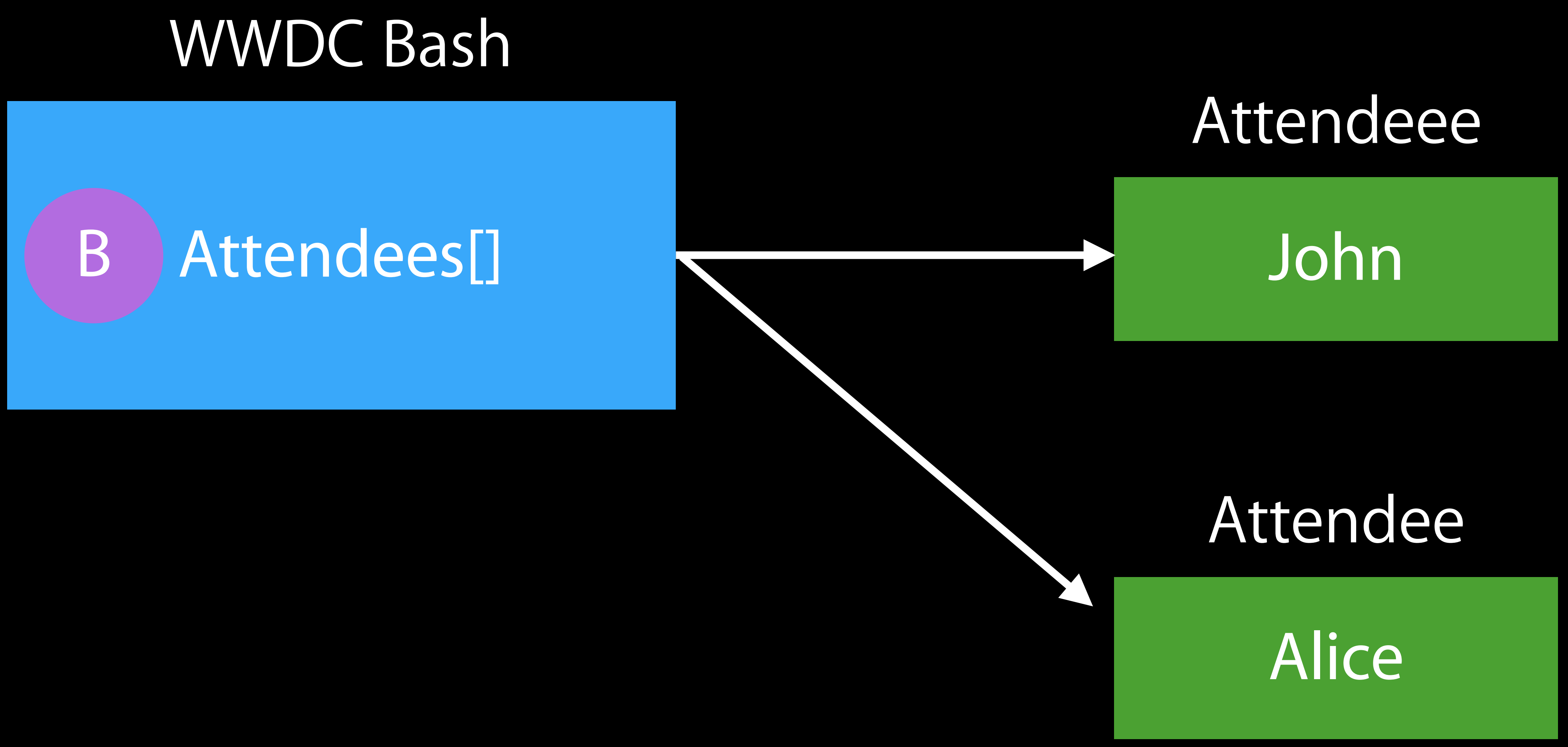
WWDC Bash

B Attendees[]

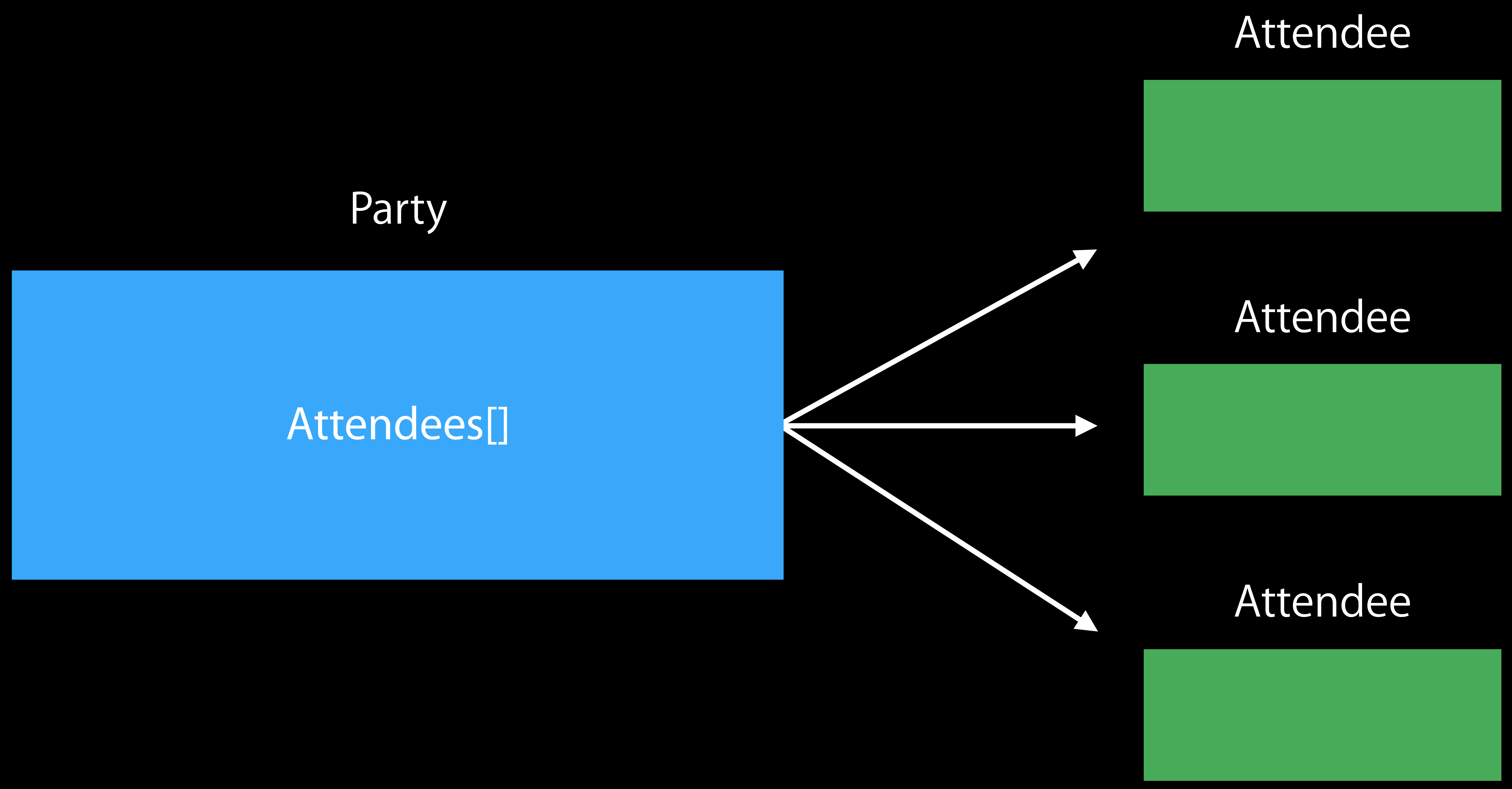
Attendeee

John

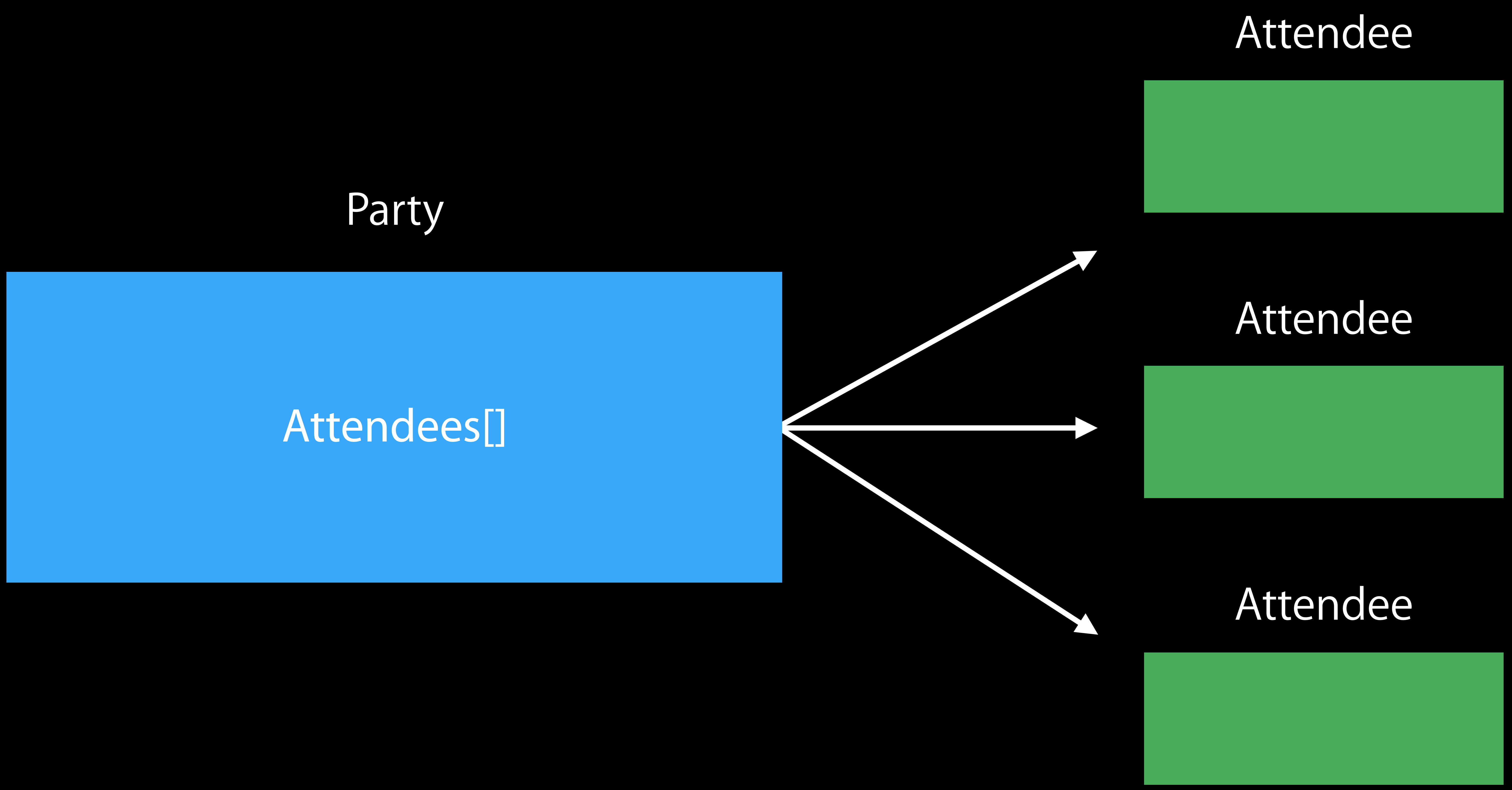
Handling Conflicts

Key	Record
ServerRecordKey	
	

Handling Conflicts

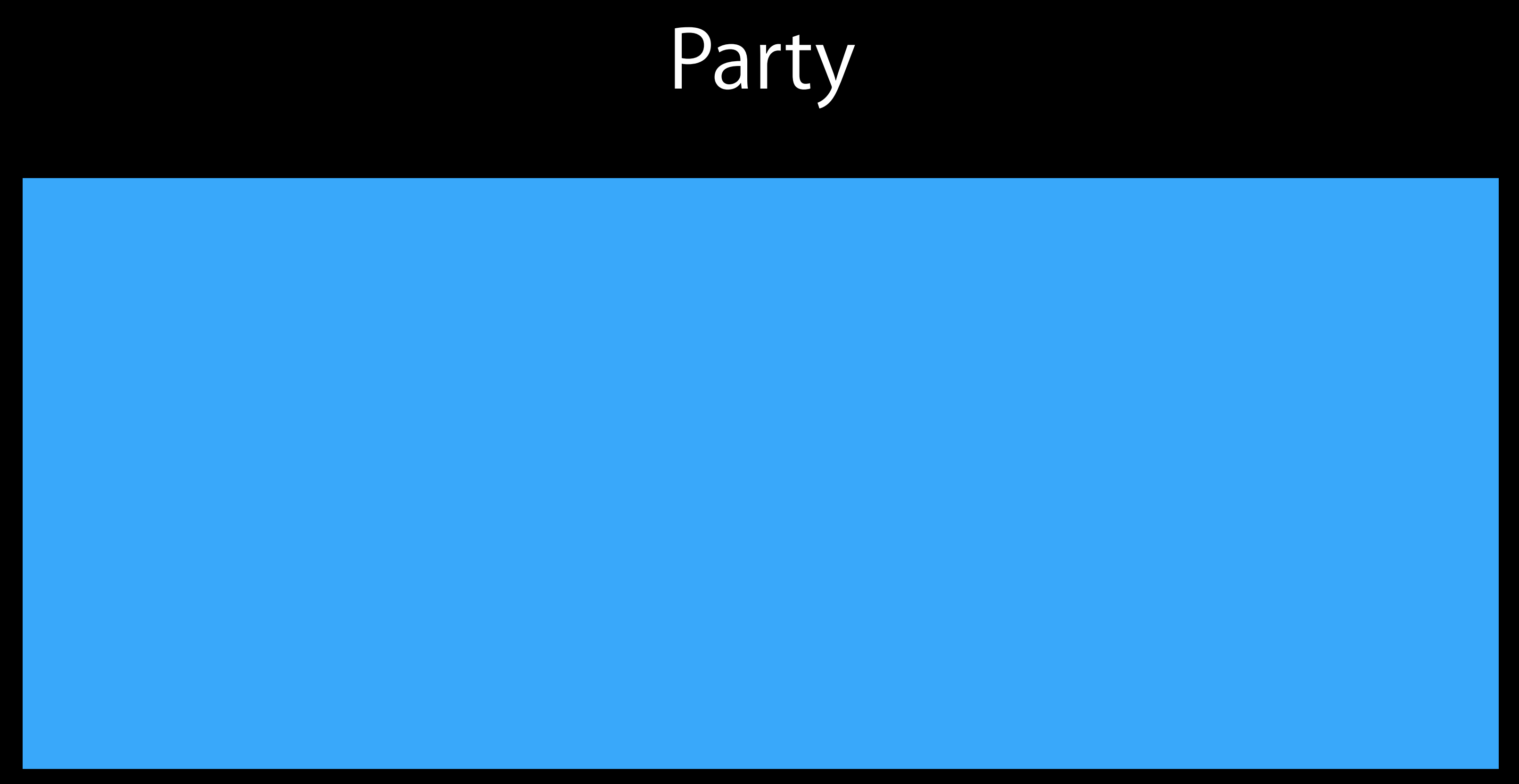


Handling Conflicts



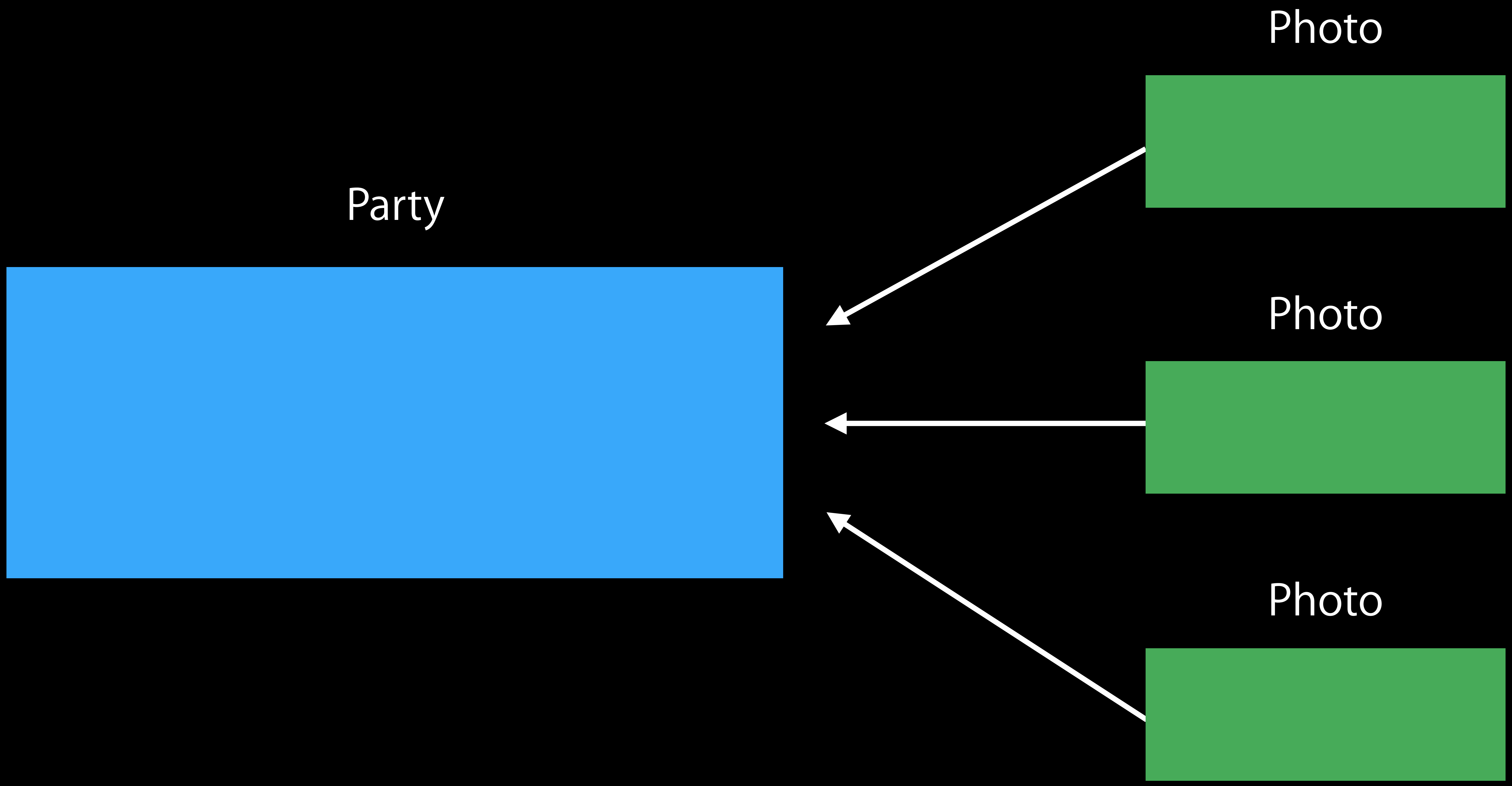
CloudKit Operations

Batch Operations



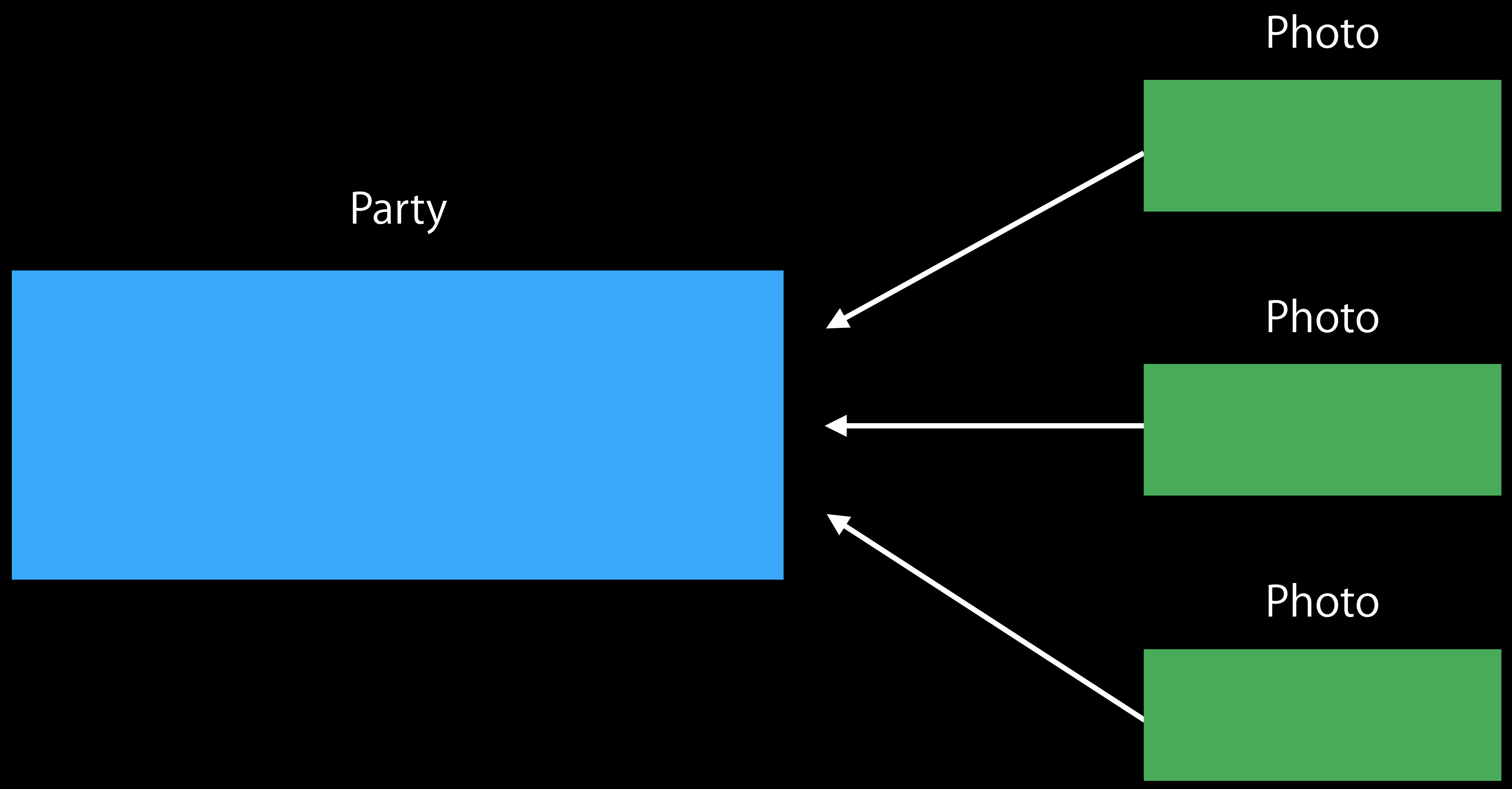
Party

Batch Operations



Batch Operations

✓

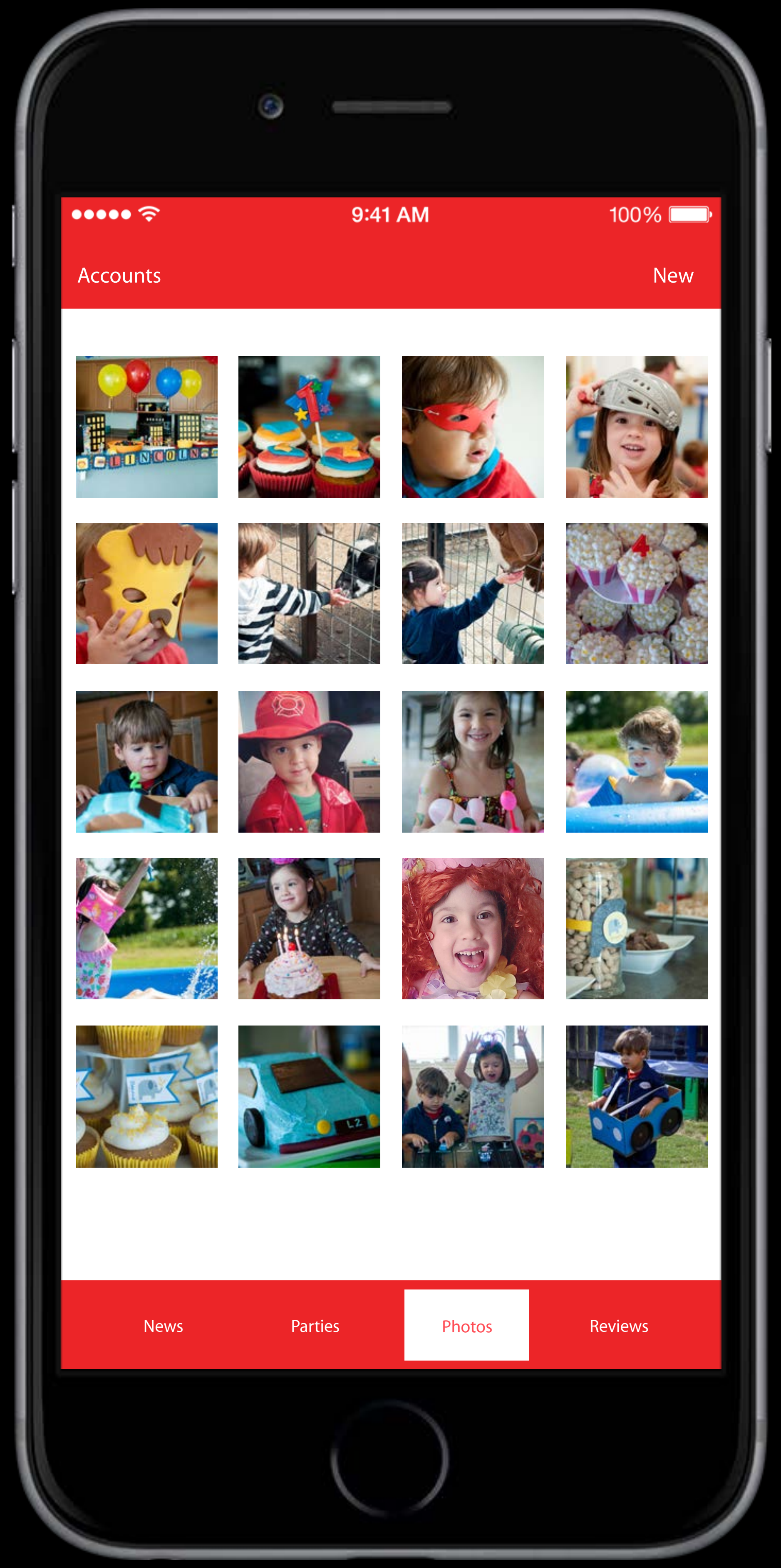


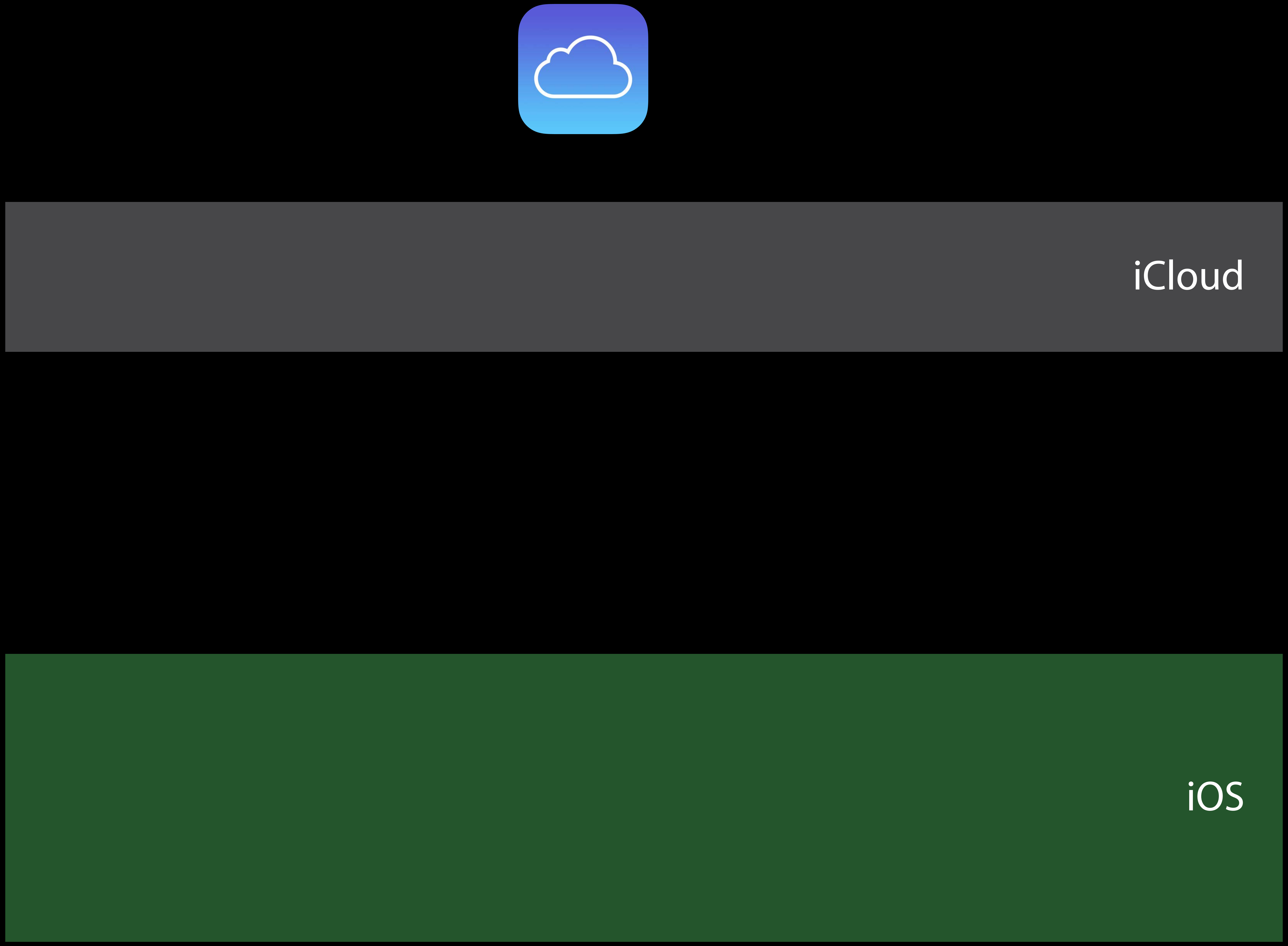
Batch Operations

Convenience API Counterpart

```
extension CKDatabase {
    func saveRecord(record: CKRecord,
         completionHandler: (CKRecord?, NSError?) -> Void)
...
}
```

Convenience API

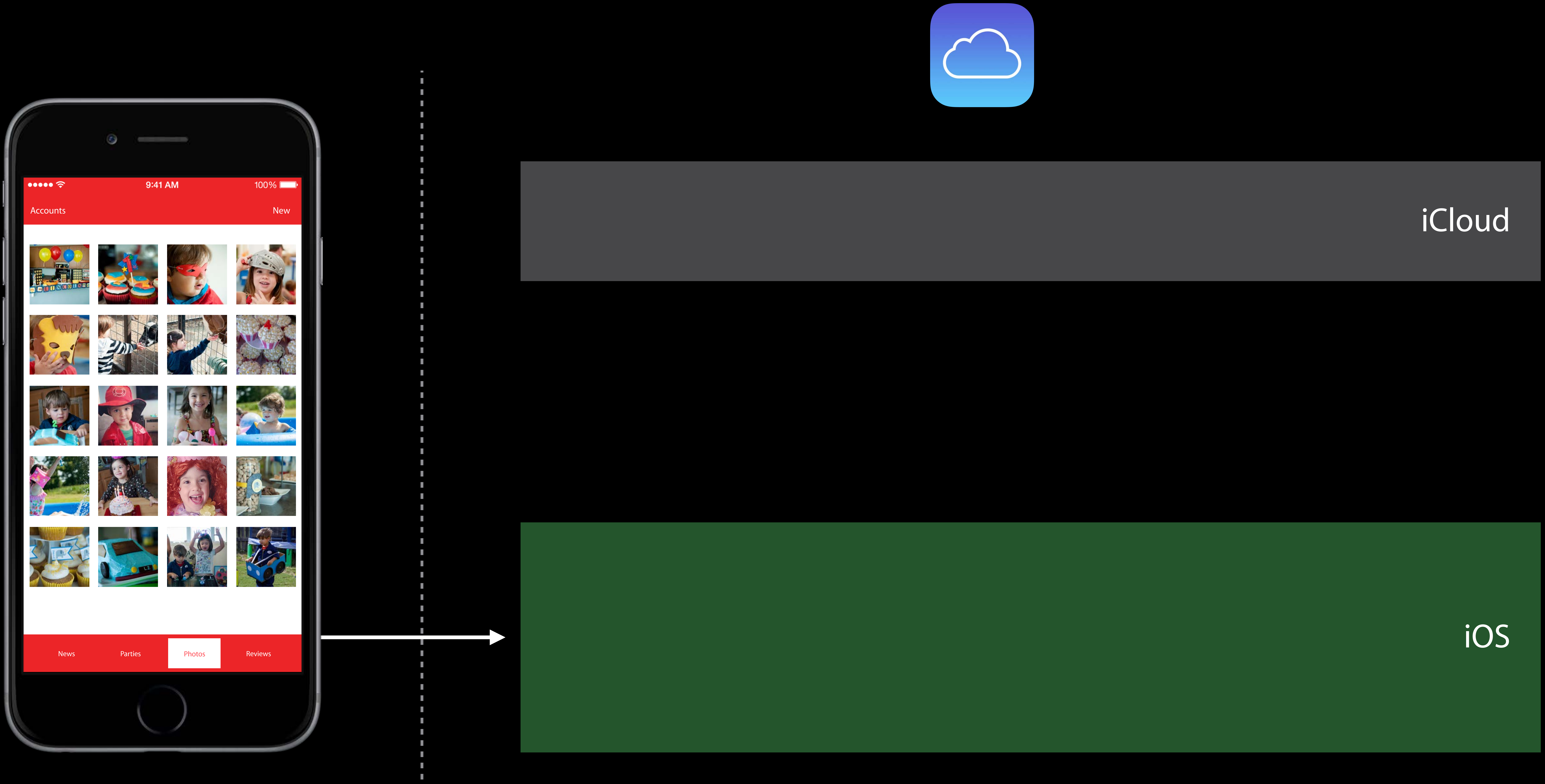




iCloud

iOS

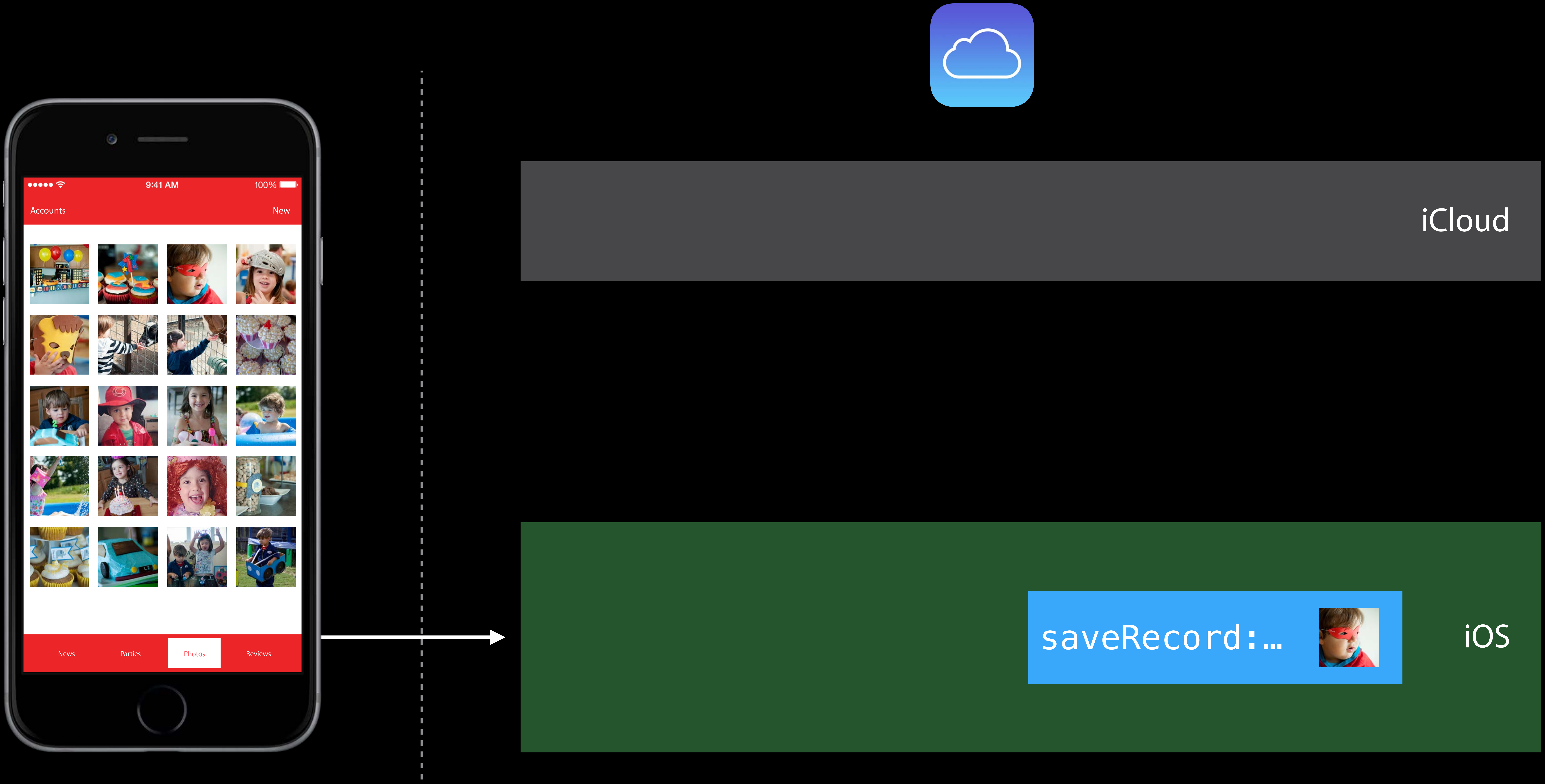
Convenience API



iCloud

iOS

Convenience API

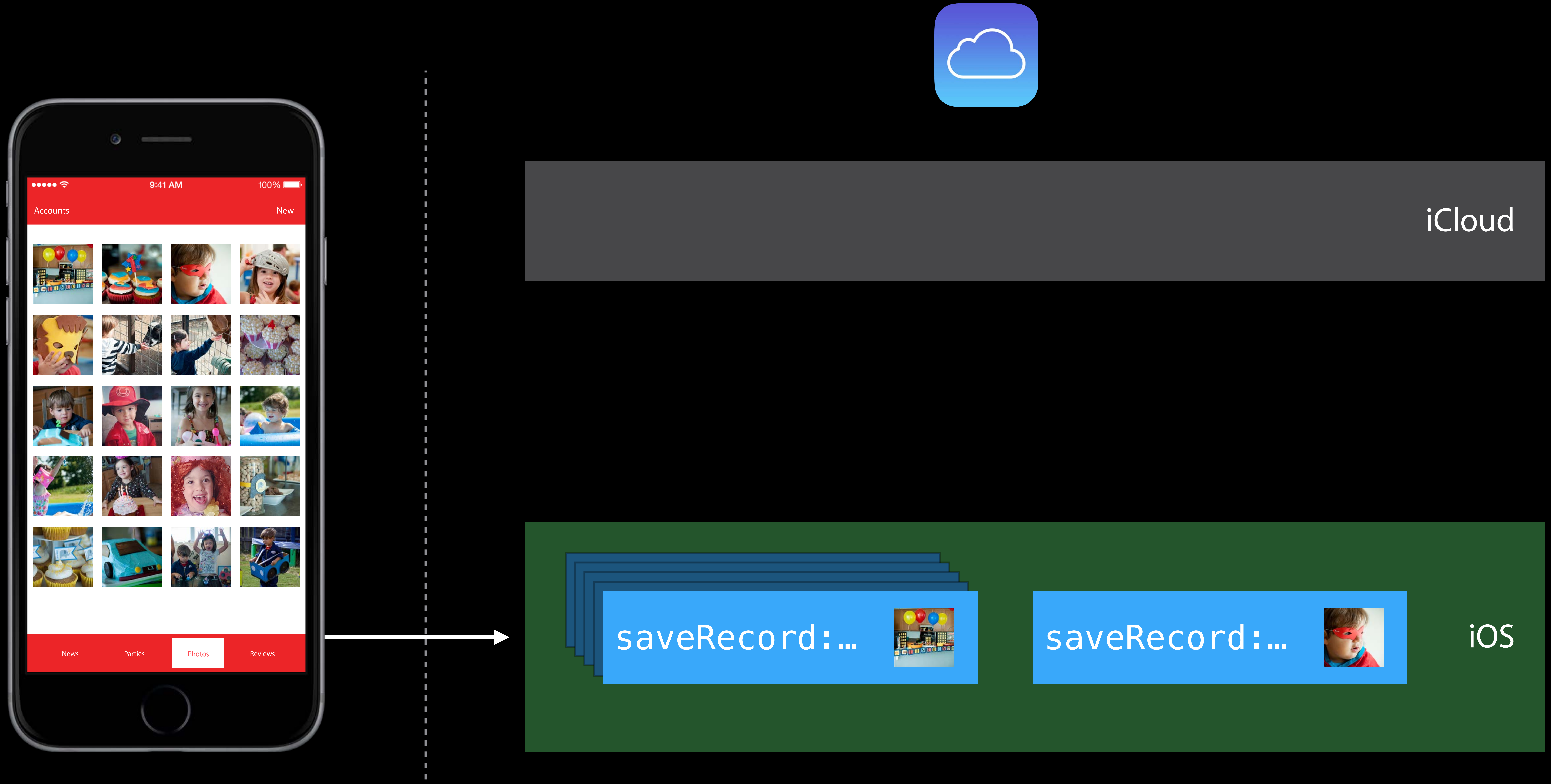


iCloud

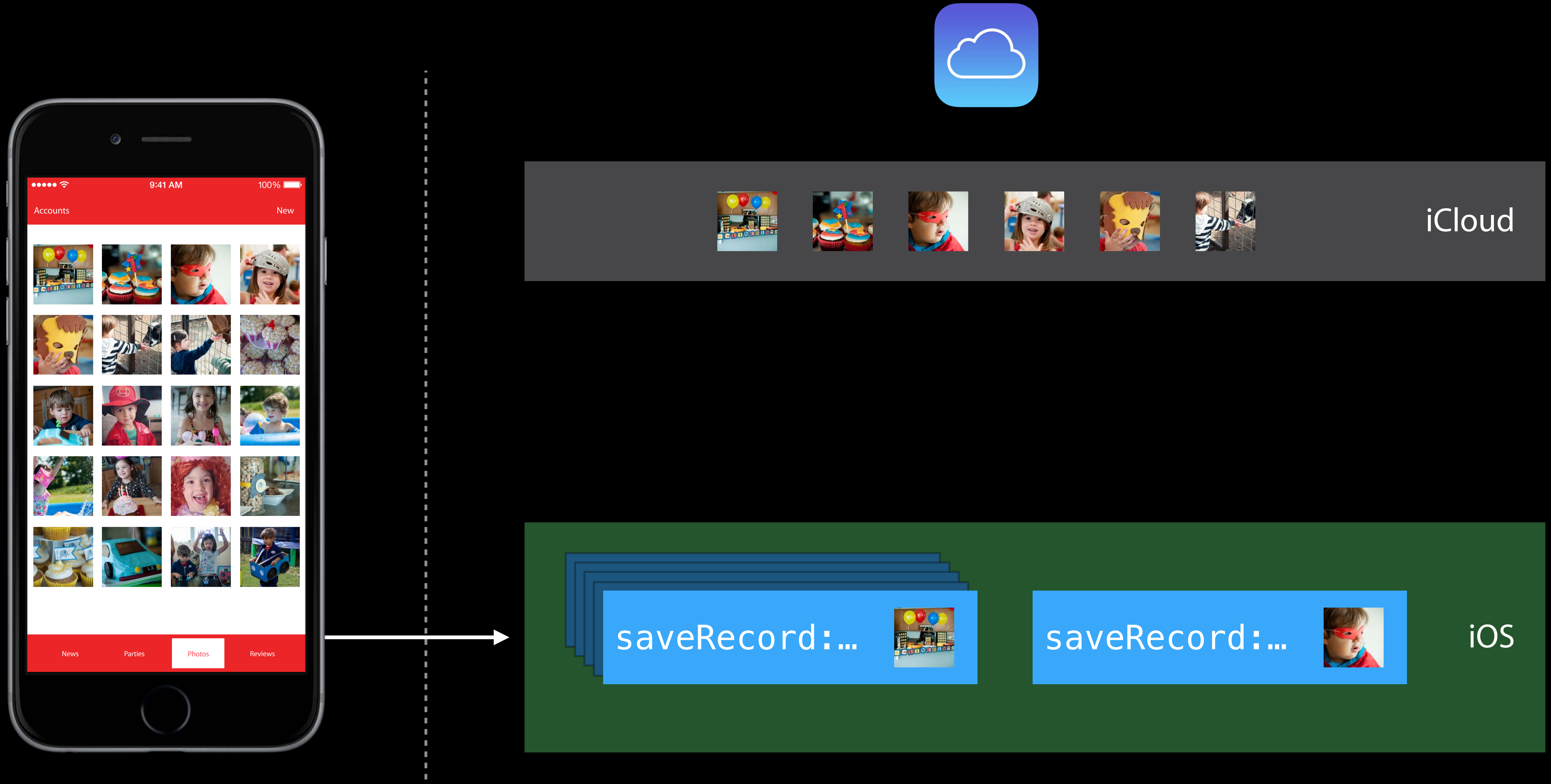
saveRecord:…

iOS

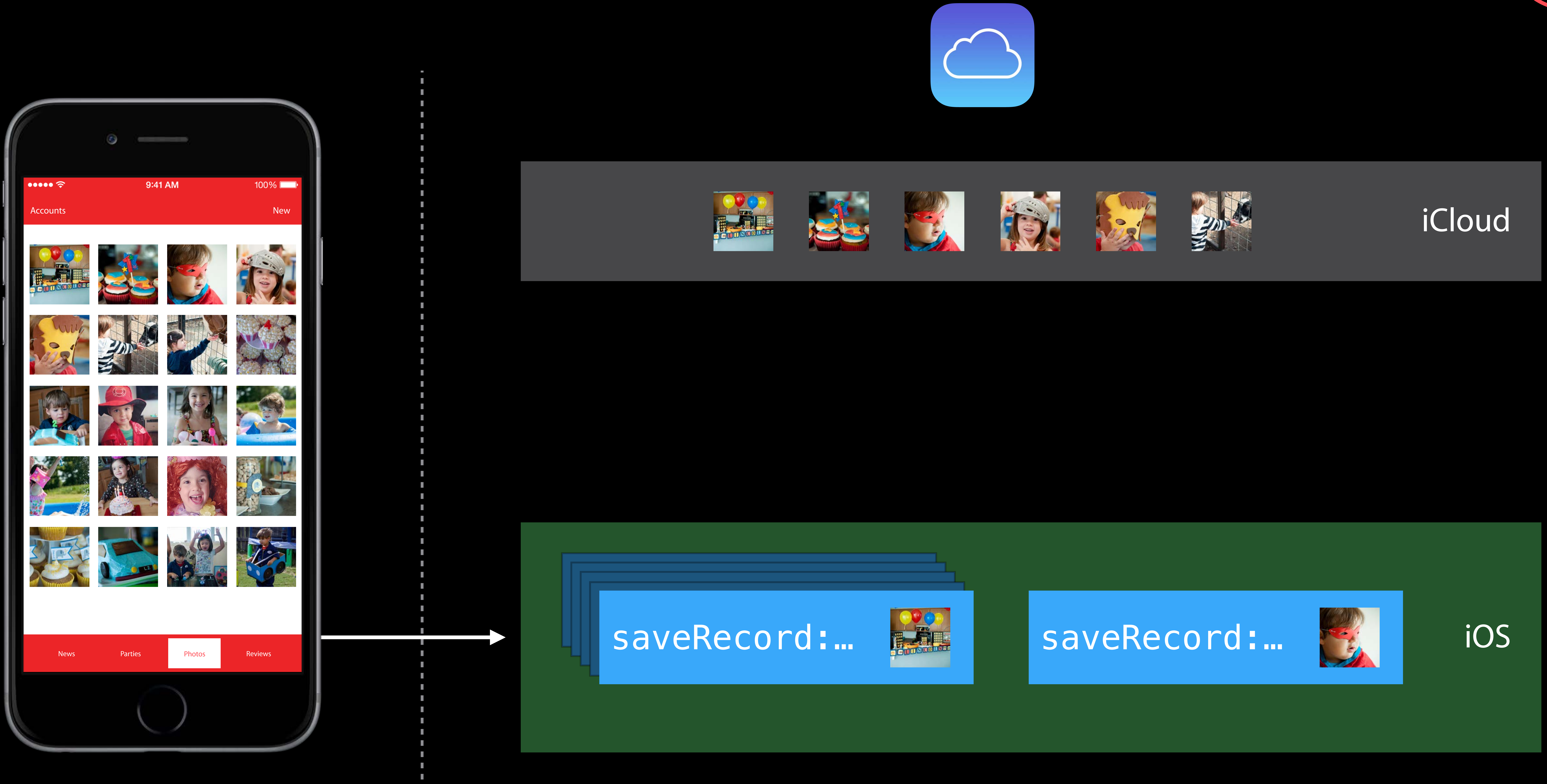
Convenience API



Convenience API



Convenience API



Batch Operations

CKOperation recap

```
extension CKDatabase {
   func saveRecord(record: CKRecord,
        completionHandler: (CKRecord?, NSError?) -> Void)
…
}
```

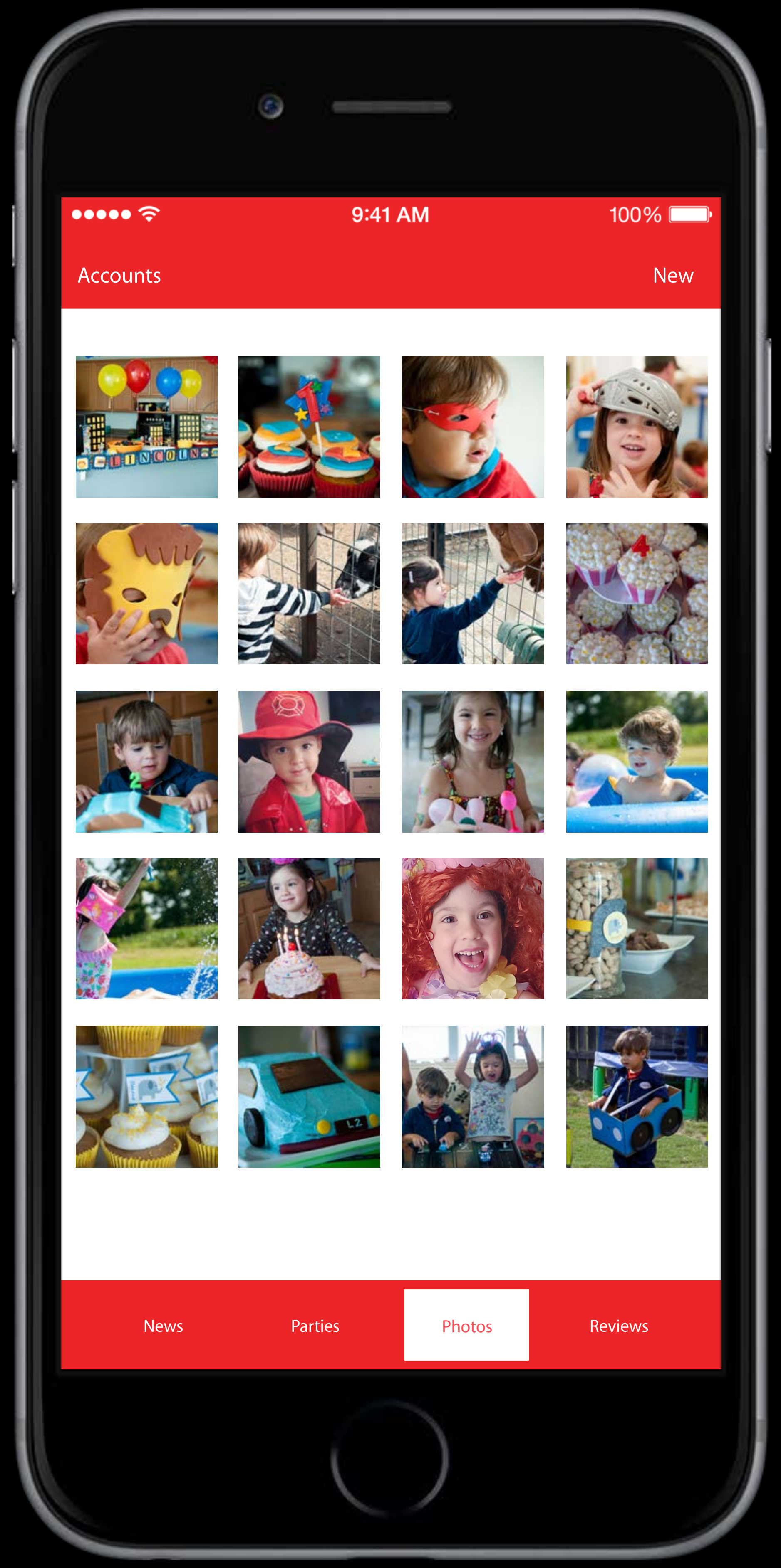
Batch Operations

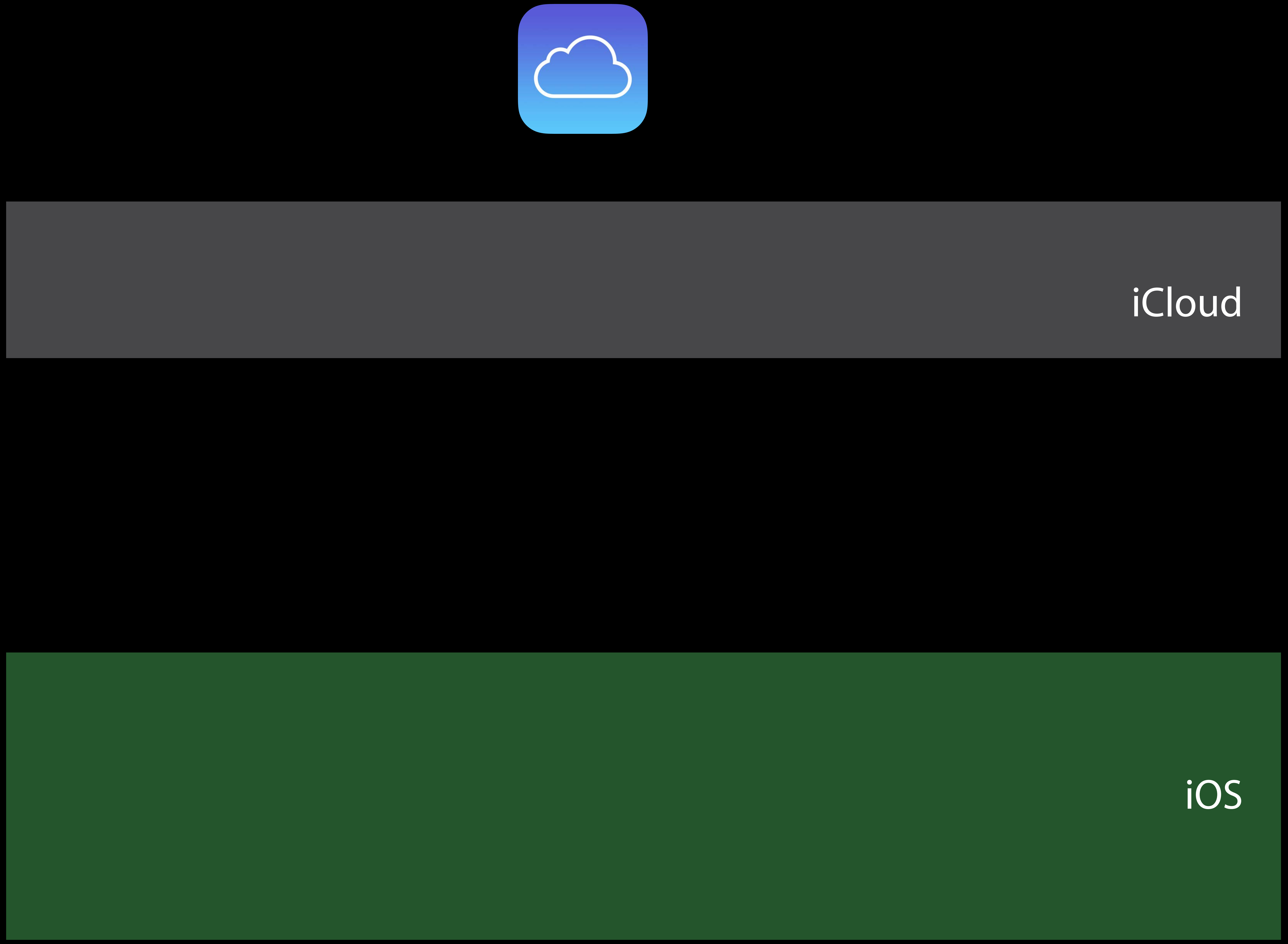
CKOperation recap

```
extension CKDatabase {
   func saveRecord(record: CKRecord,
        completionHandler: (CKRecord?, NSError?) -> Void)
…
}
```

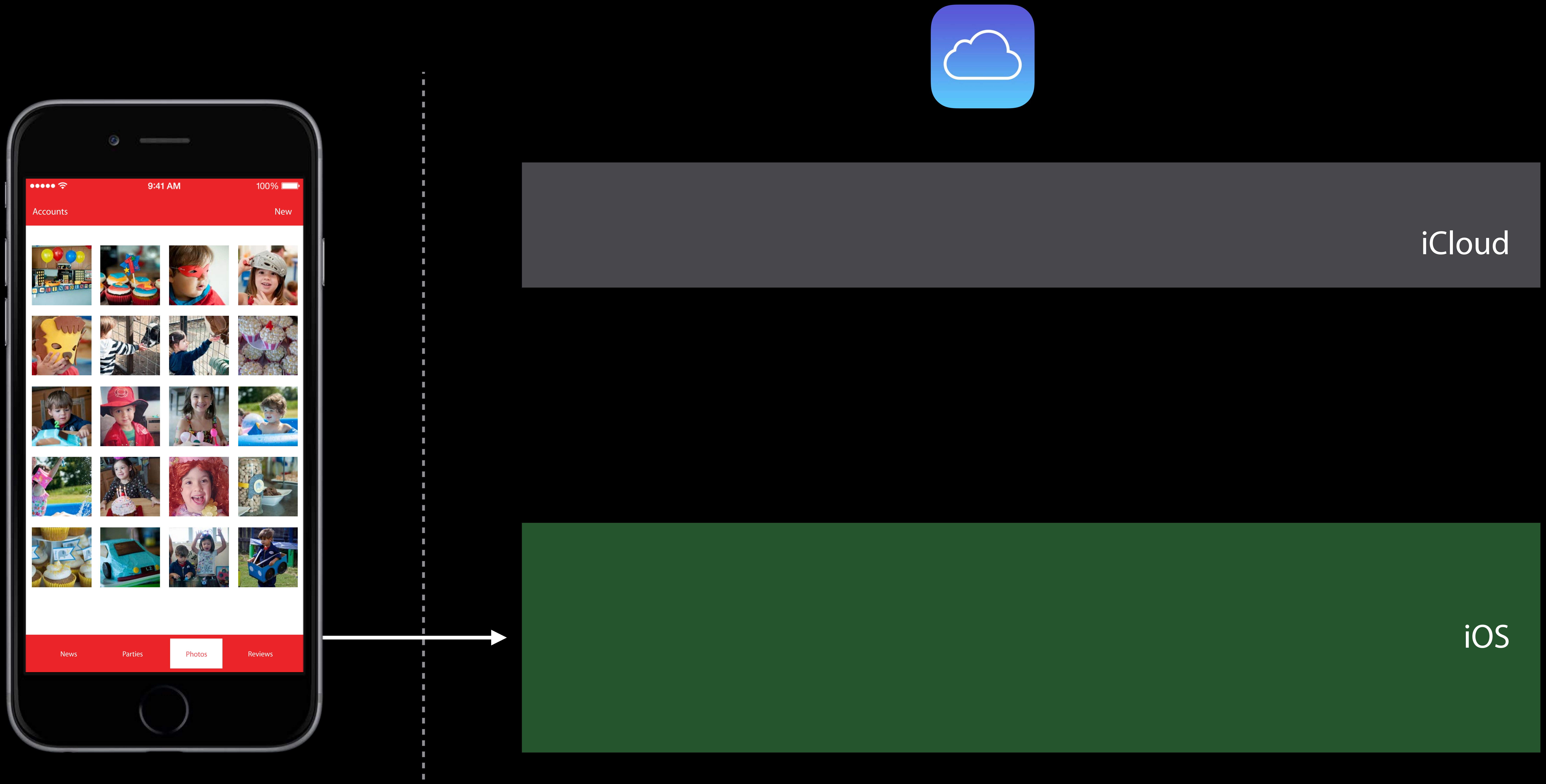
```
class CKModifyRecordsOperation : CKDatabaseOperation {
   init()
   convenience init(recordsToSave: [CKRecord]?,
                 recordIDsToDelete: [CKRecordID]?)
…
}
```

Batch Operations

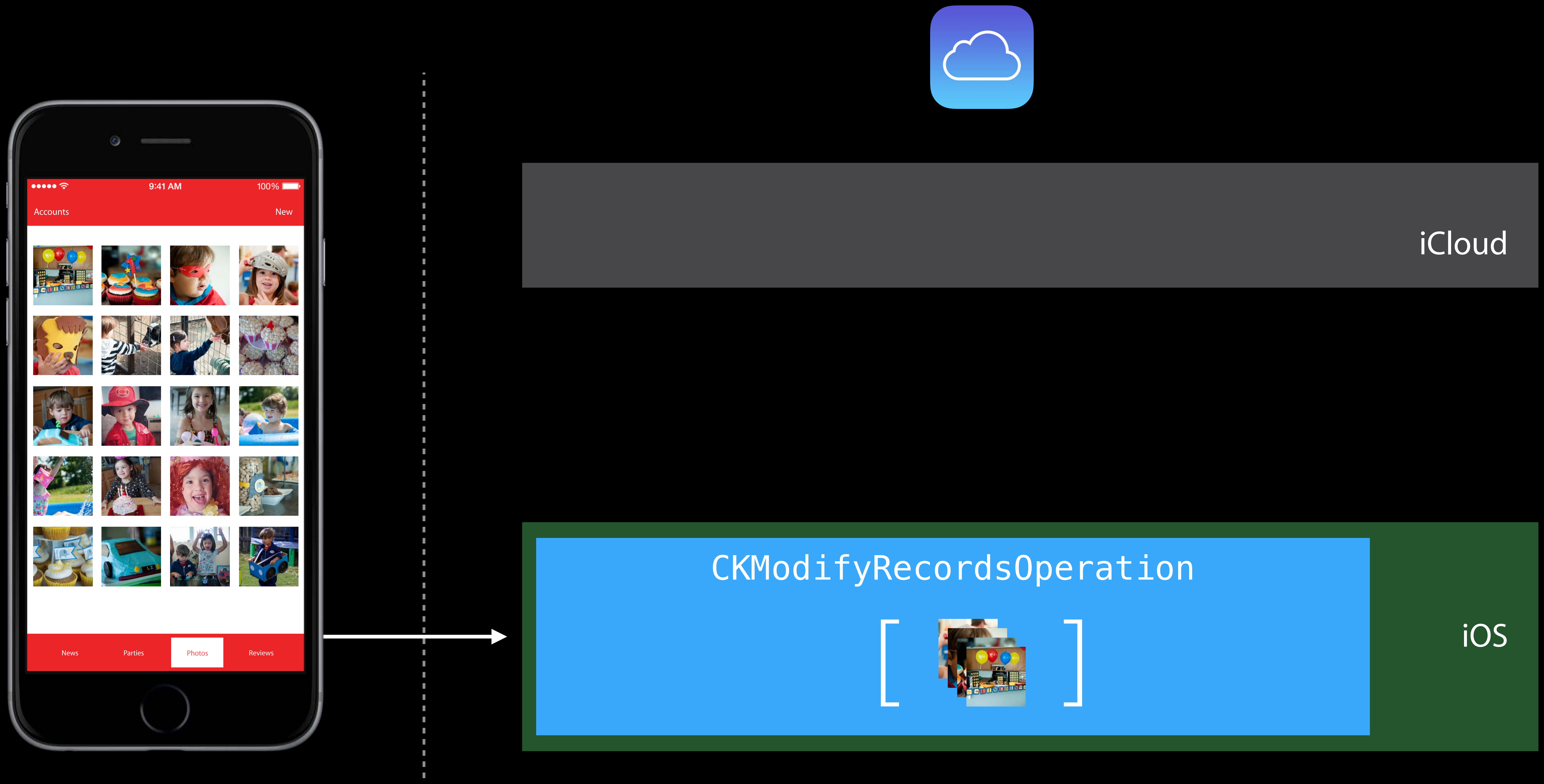




Batch Operations



Batch Operations



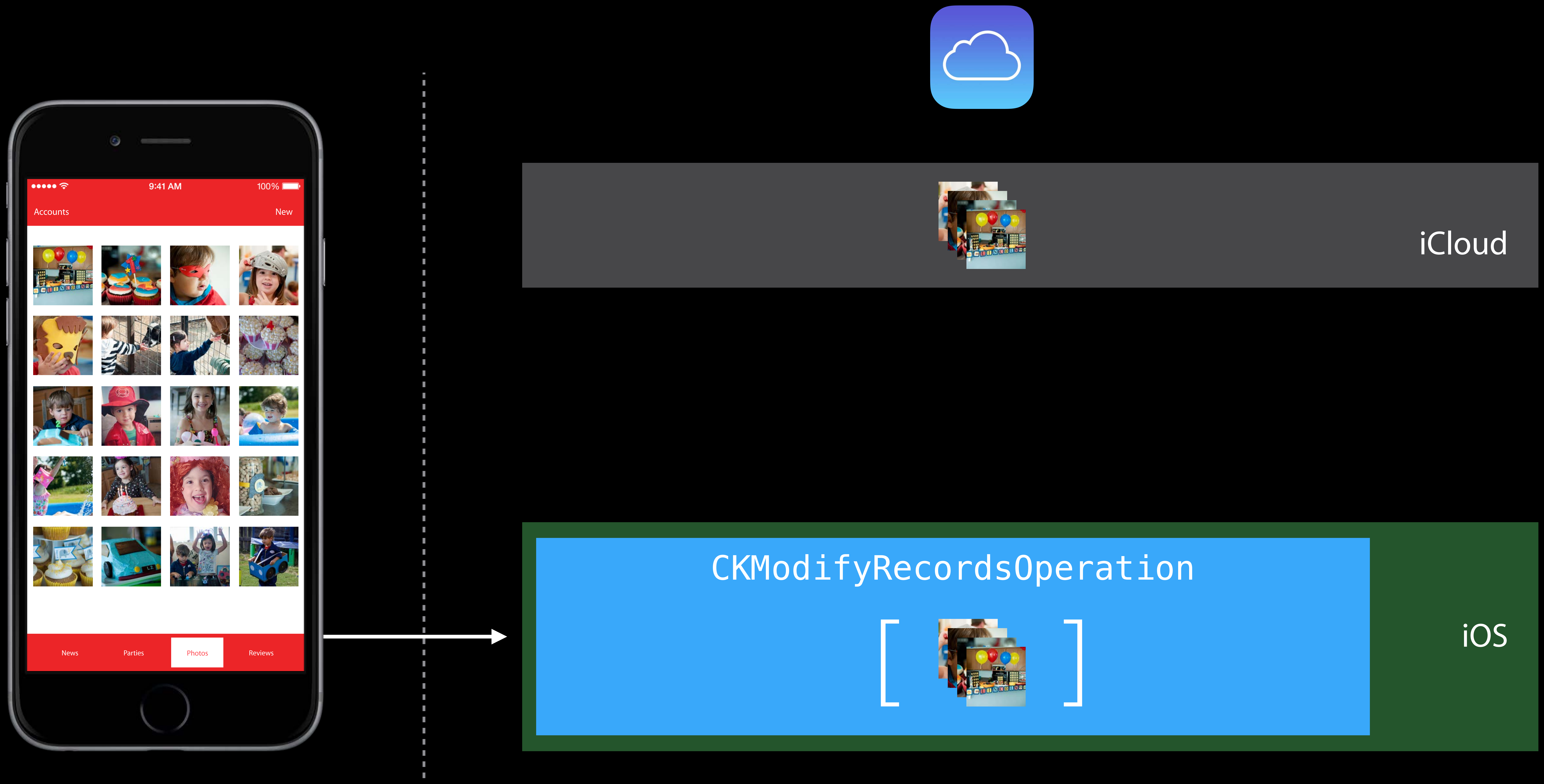
iCloud

CKModifyRecordsOperation

[ ]

iOS

Batch Operations



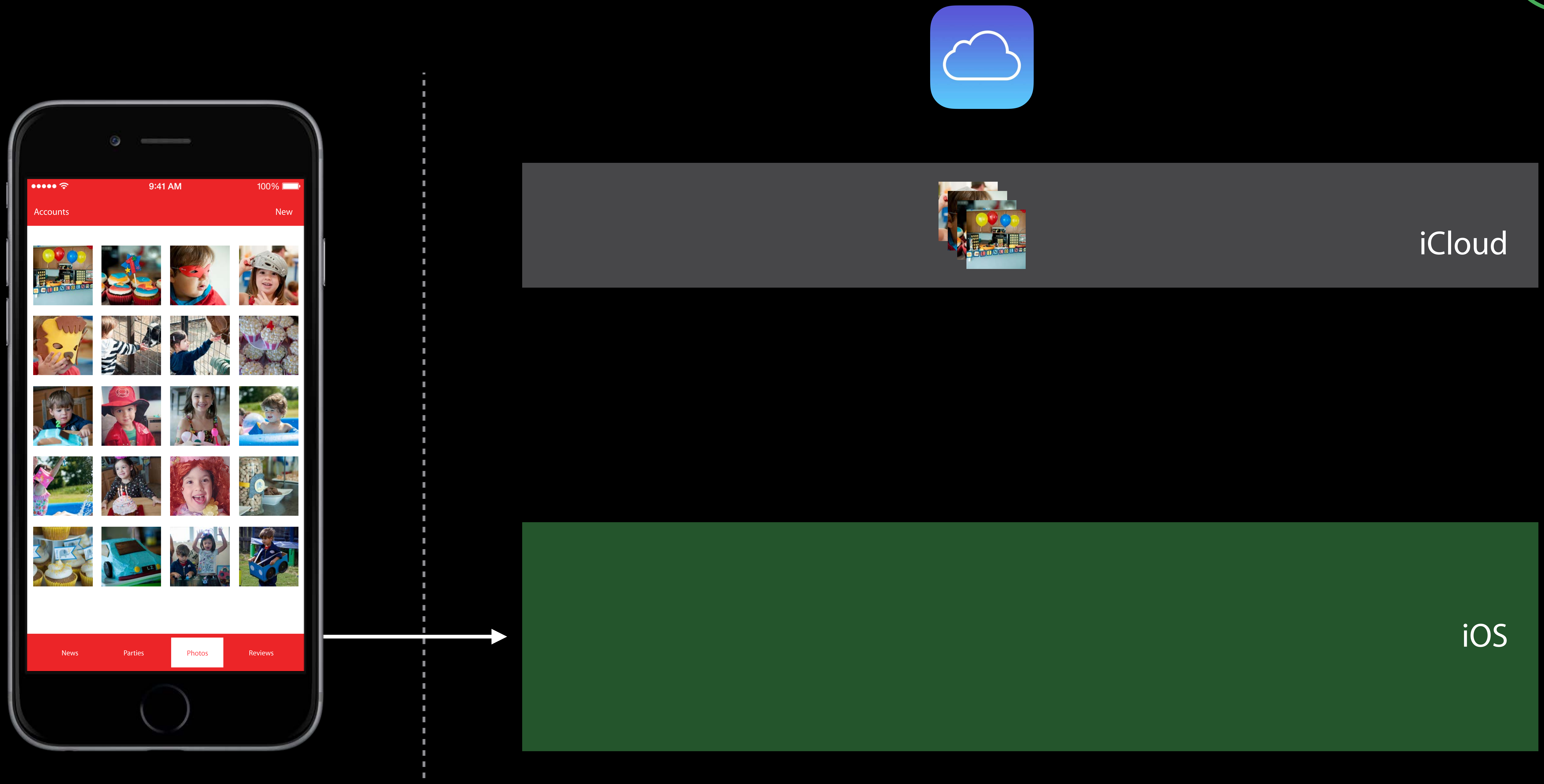
iCloud

CKModifyRecordsOperation

[ ]

iOS

Batch Operations



Batch Operations

Batch size limits

Batch Operations

Batch size limits

Server imposed limits

Batch Operations

Batch size limits

Server imposed limits

- Number of items per request

Batch Operations

Batch size limits

Server imposed limits

- Number of items per request
- Size of request

Batch Operations

Batch size limits

Server imposed limits

- Number of items per request
- Size of request

`CKErrorLimitExceeded`

Batch Operations

Partial errors

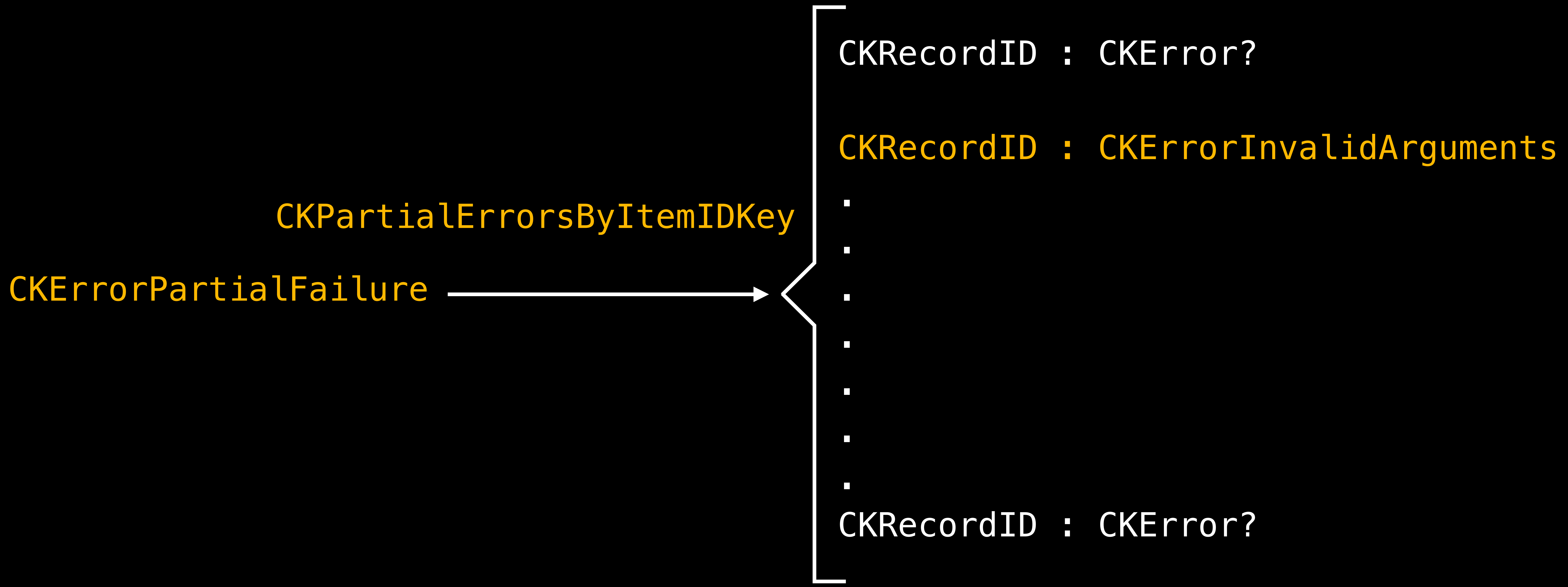
Batch Operations

Partial errors

```
CKErrorPartialFailure
```

Batch Operations

Partial errors



CKPartialErrorsByItemIDKey

CKErrorPartialFailure

CKRecordID : CKError?

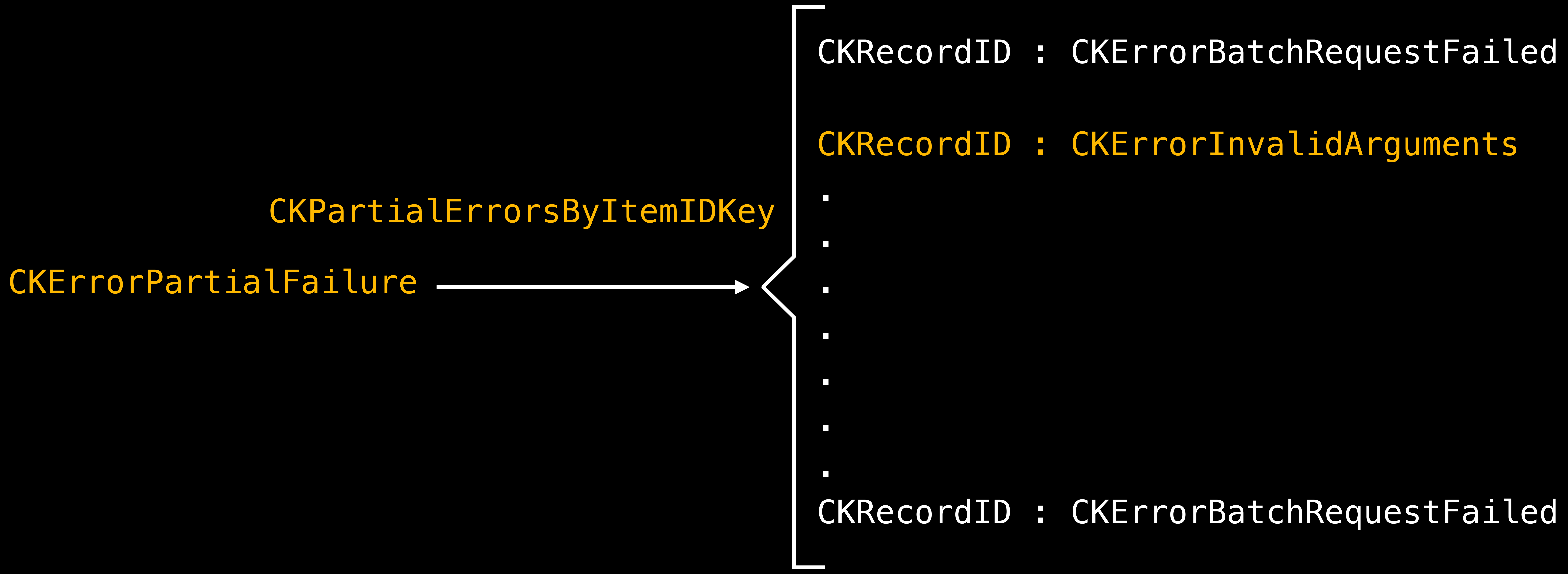
CKRecordID : CKErrorInvalidArguments

.
.
.
.
.
.
.

CKRecordID : CKError?

Batch Operations

Partial errors with atomic updates



CKPartialErrorsByItemIDKey

CKErrorPartialFailure

CKRecordID : CKErrorBatchRequestFailed

CKRecordID : CKErrorInvalidArguments

.
.
.
.
.
.
.

CKRecordID : CKErrorBatchRequestFailed

Queries

Queries

In ClownCentral

Queries

In ClownCentral

```
let query = CKQuery(recordType: "Photo",
               predicate: NSPredicate(format: "Party == %@", partyRecordID))
```

Queries

In ClownCentral

```
let query = CKQuery(recordType: "Photo",
               predicate: NSPredicate(format: "Party == %@", partyRecordID))
```

Parties may have a lot of photos

Queries

In ClownCentral

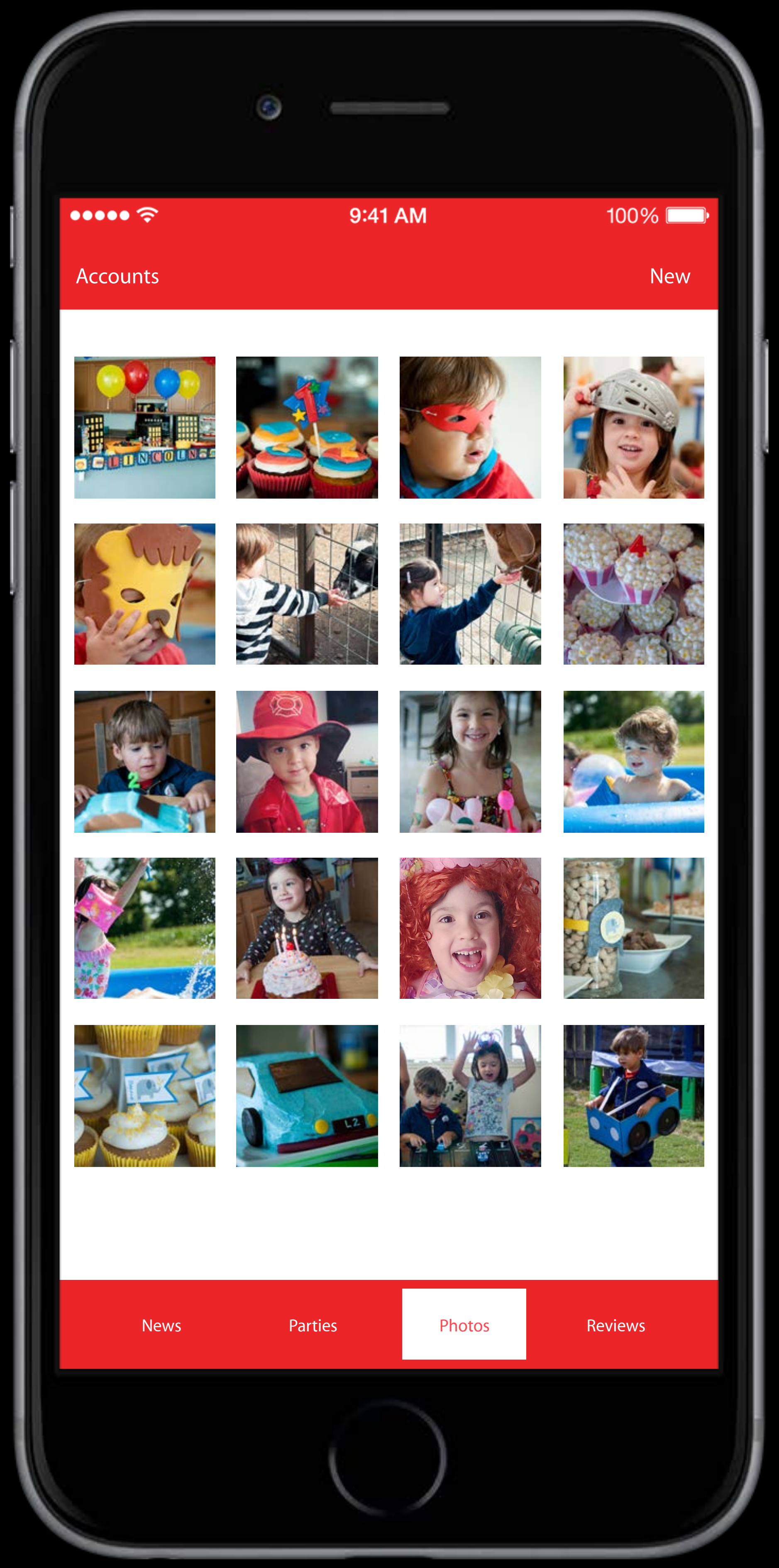
```
let query = CKQuery(recordType: "Photo",
               predicate: NSPredicate(format: "Party == %@", partyRecordID))
```

Parties may have a lot of photos

Optimize download with `CKQueryOperation`

CKQueryOperation

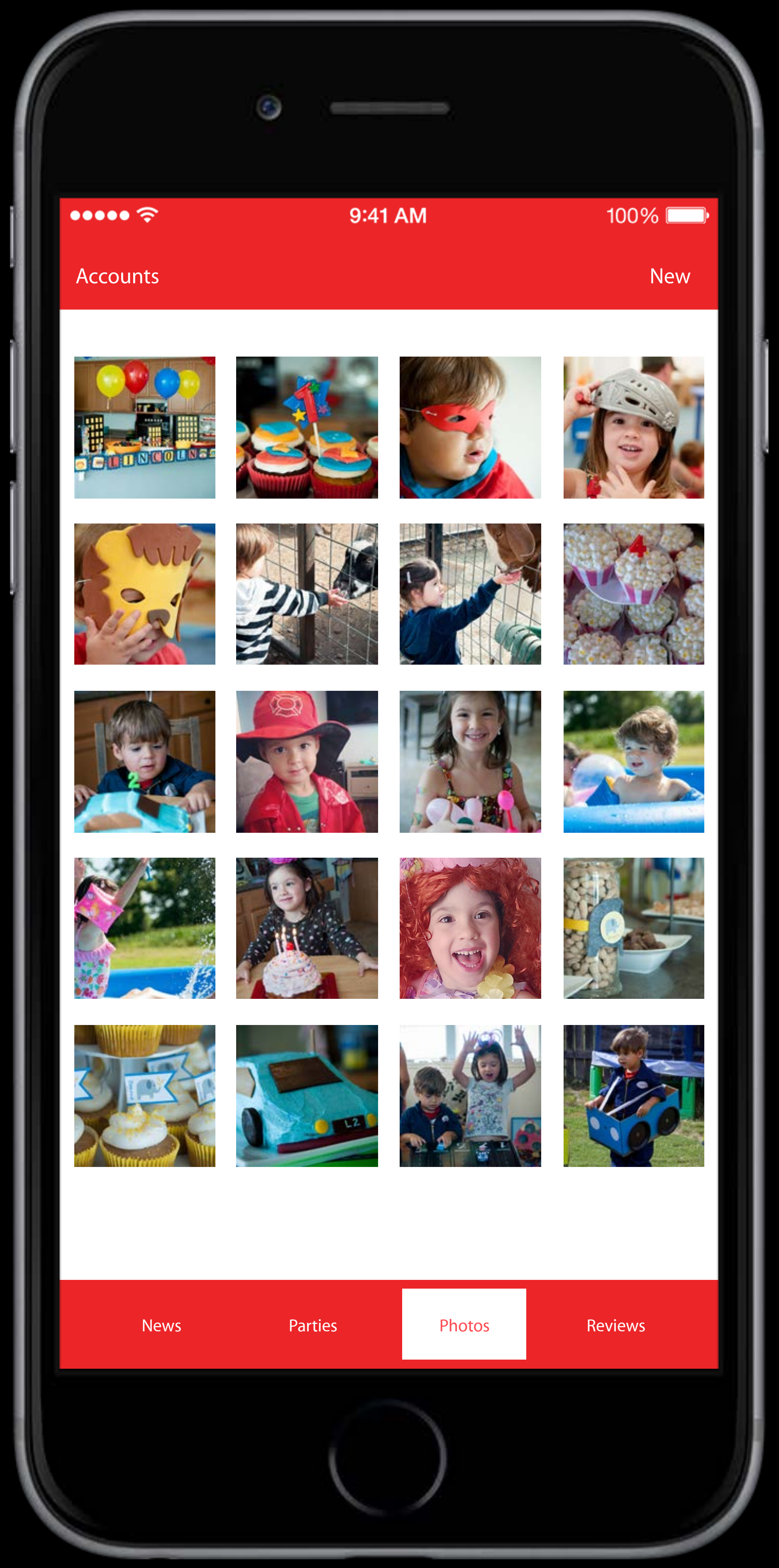
How many photos do we need?



CKQueryOperation

How many photos do we need?

```
let queryOperation = CKQueryOperation…
queryOperation.resultsLimit = 20
```

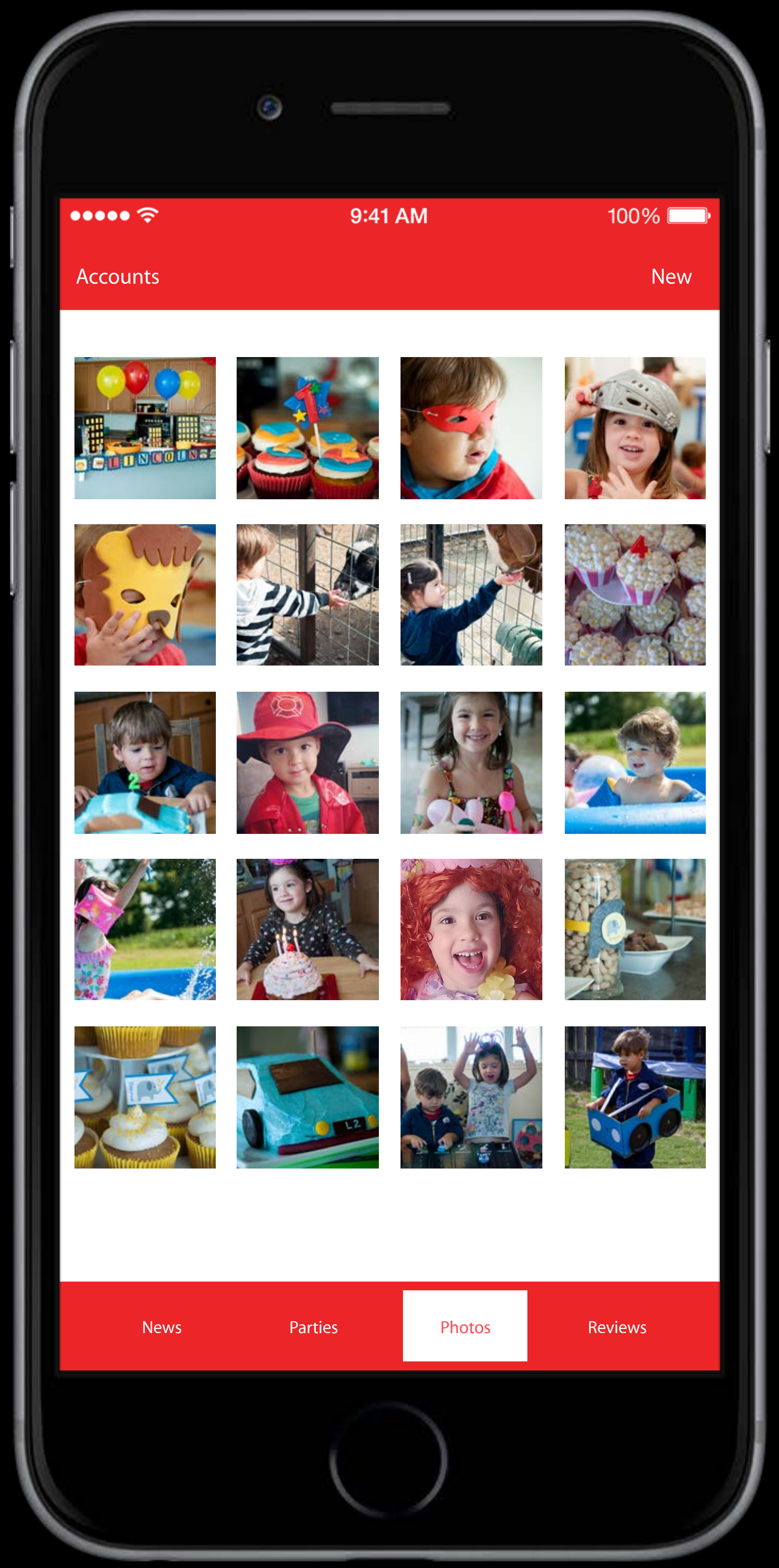


CKQueryOperation

How many photos do we need?

```
let queryOperation = CKQueryOperation…
queryOperation.resultsLimit = 20
```

Also available on:



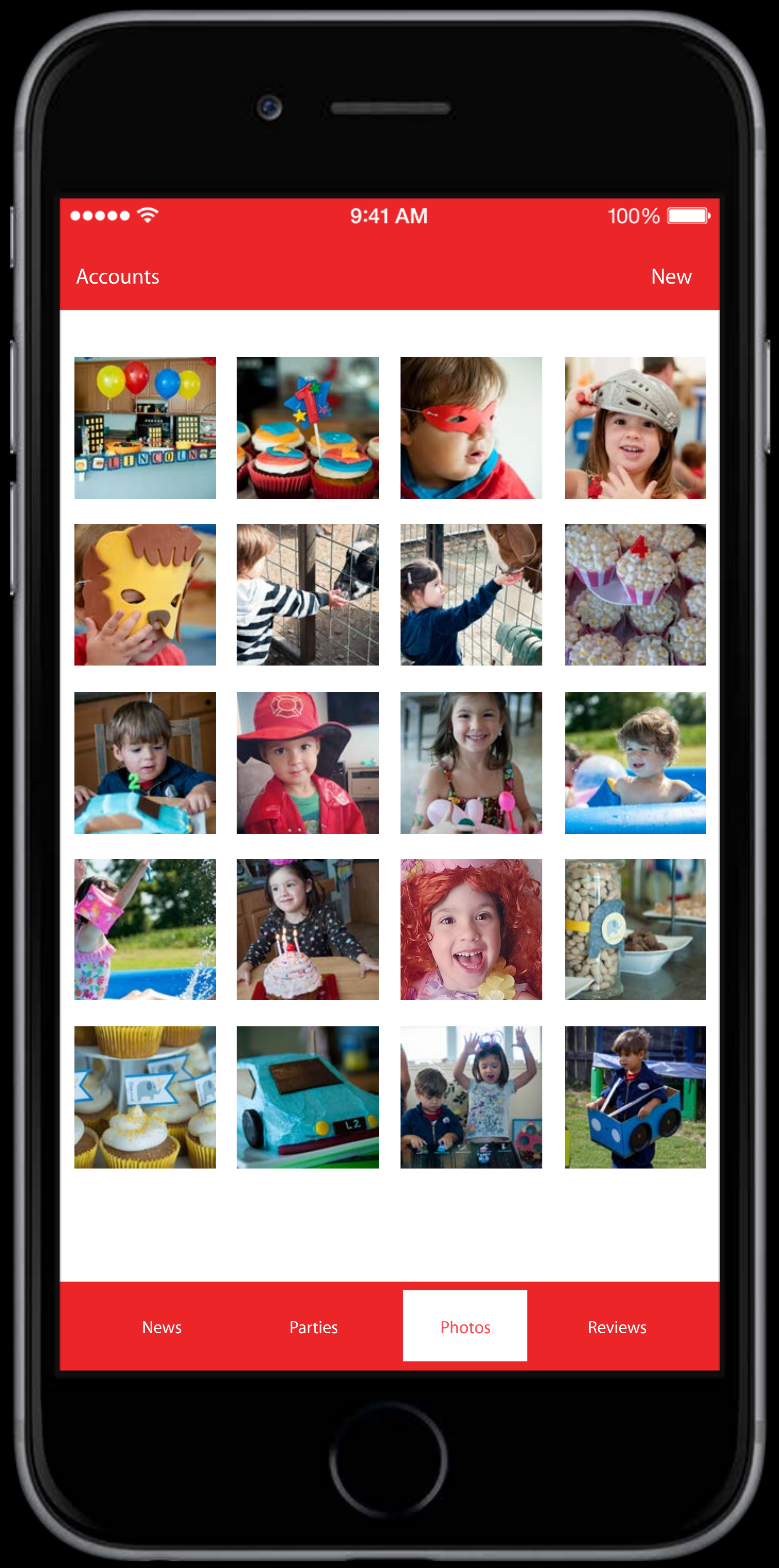
CKQueryOperation

How many photos do we need?

```
let queryOperation = CKQueryOperation…
queryOperation.resultsLimit = 20
```

Also available on:

- `CKFetchRecordChangesOperation`



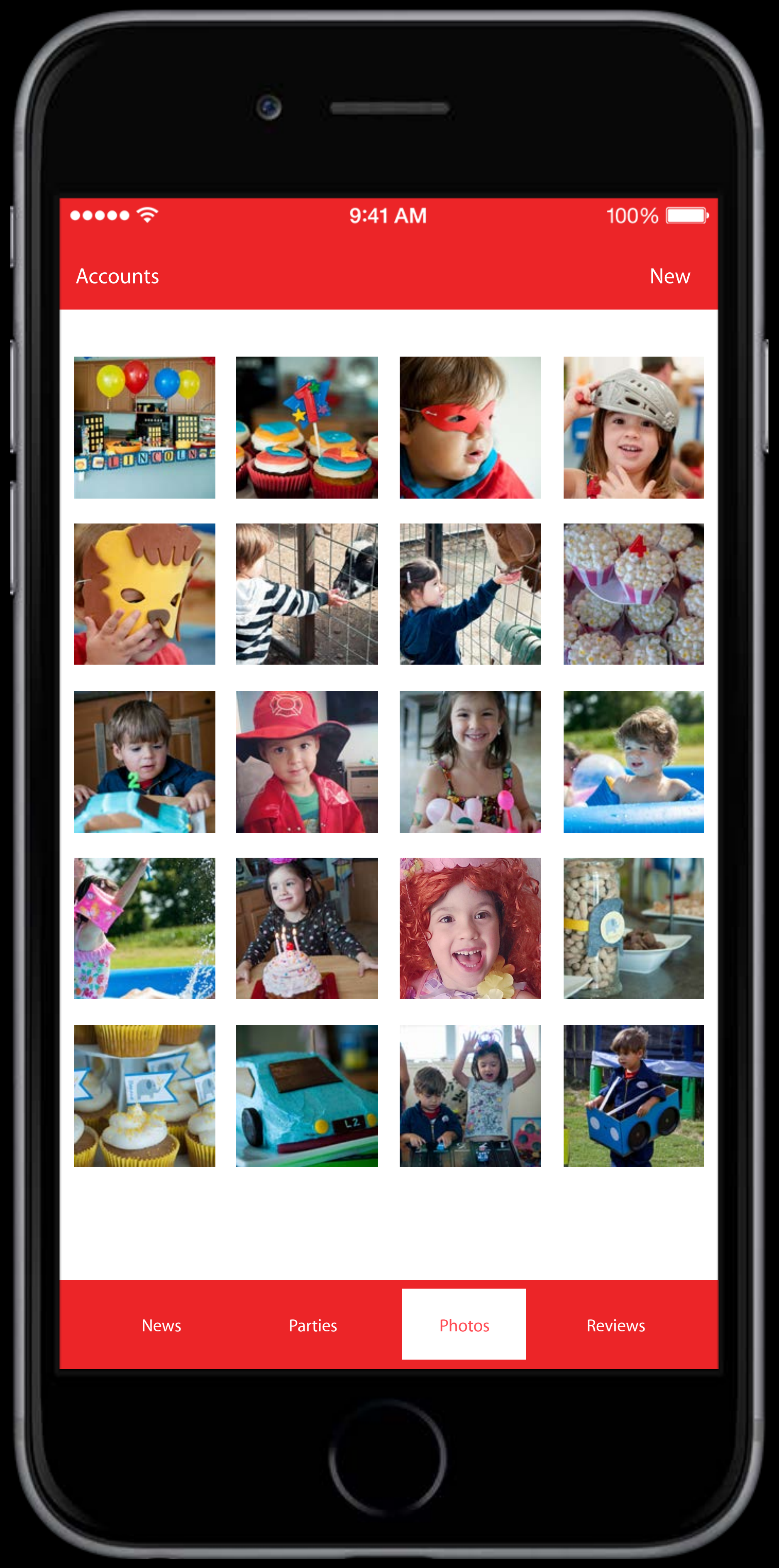
CKQueryOperation

How many photos do we need?

```
let queryOperation = CKQueryOperation…
queryOperation.resultsLimit = 20
```

Also available on:

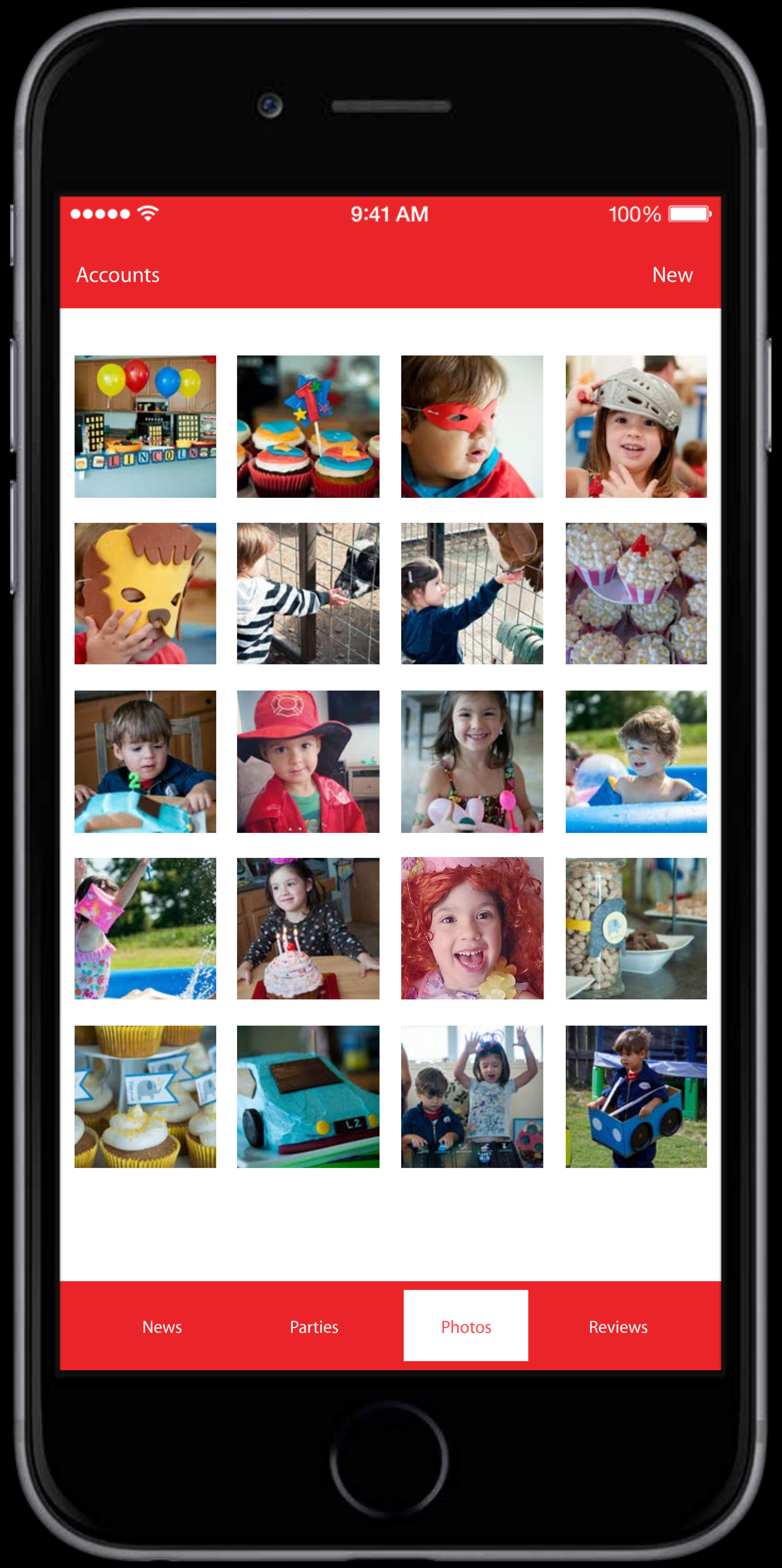
- `CKFetchRecordChangesOperation`
- `CKFetchNotificationChangesOperation`



CKQueryOperation

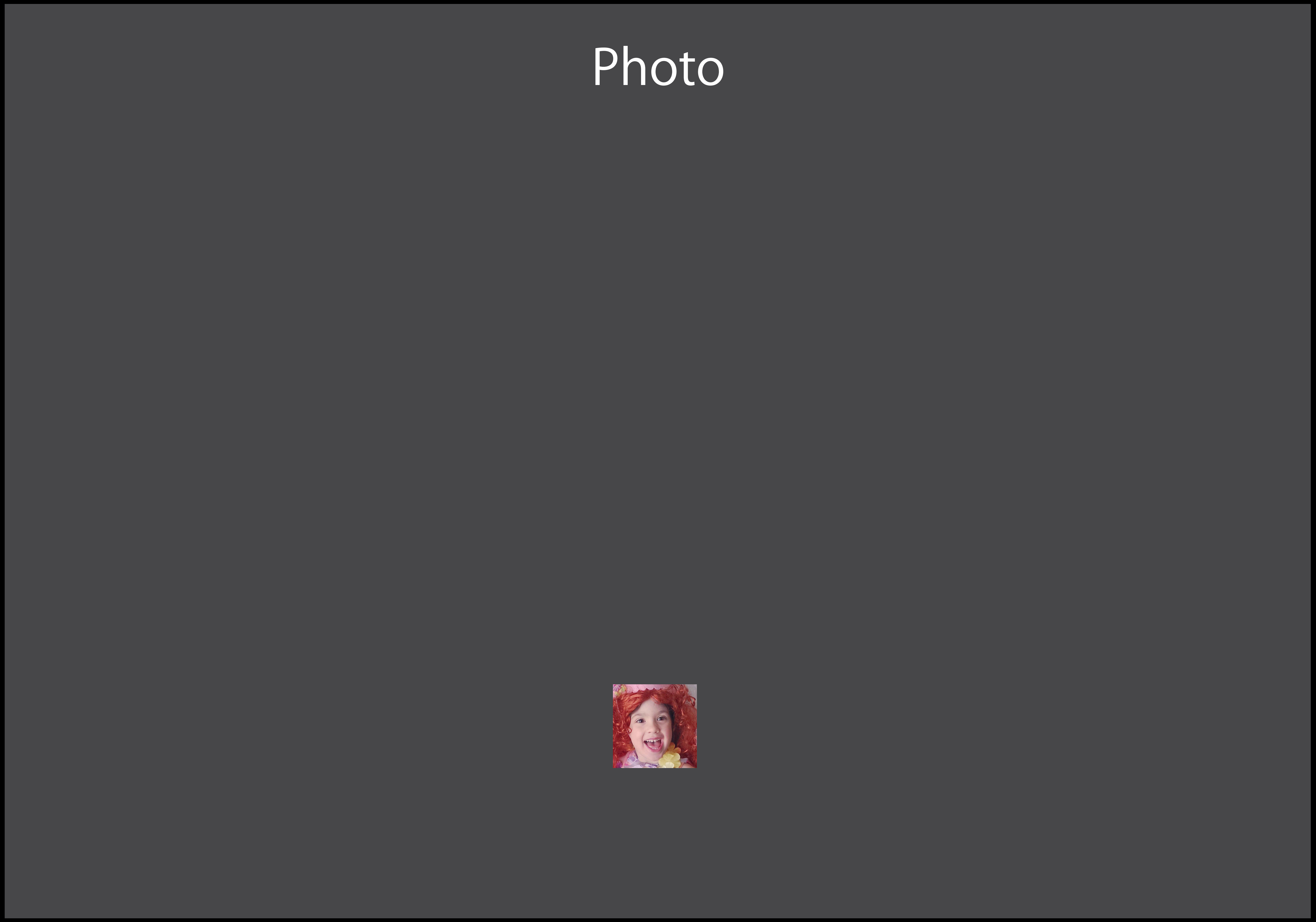
What kind of photo do we need?

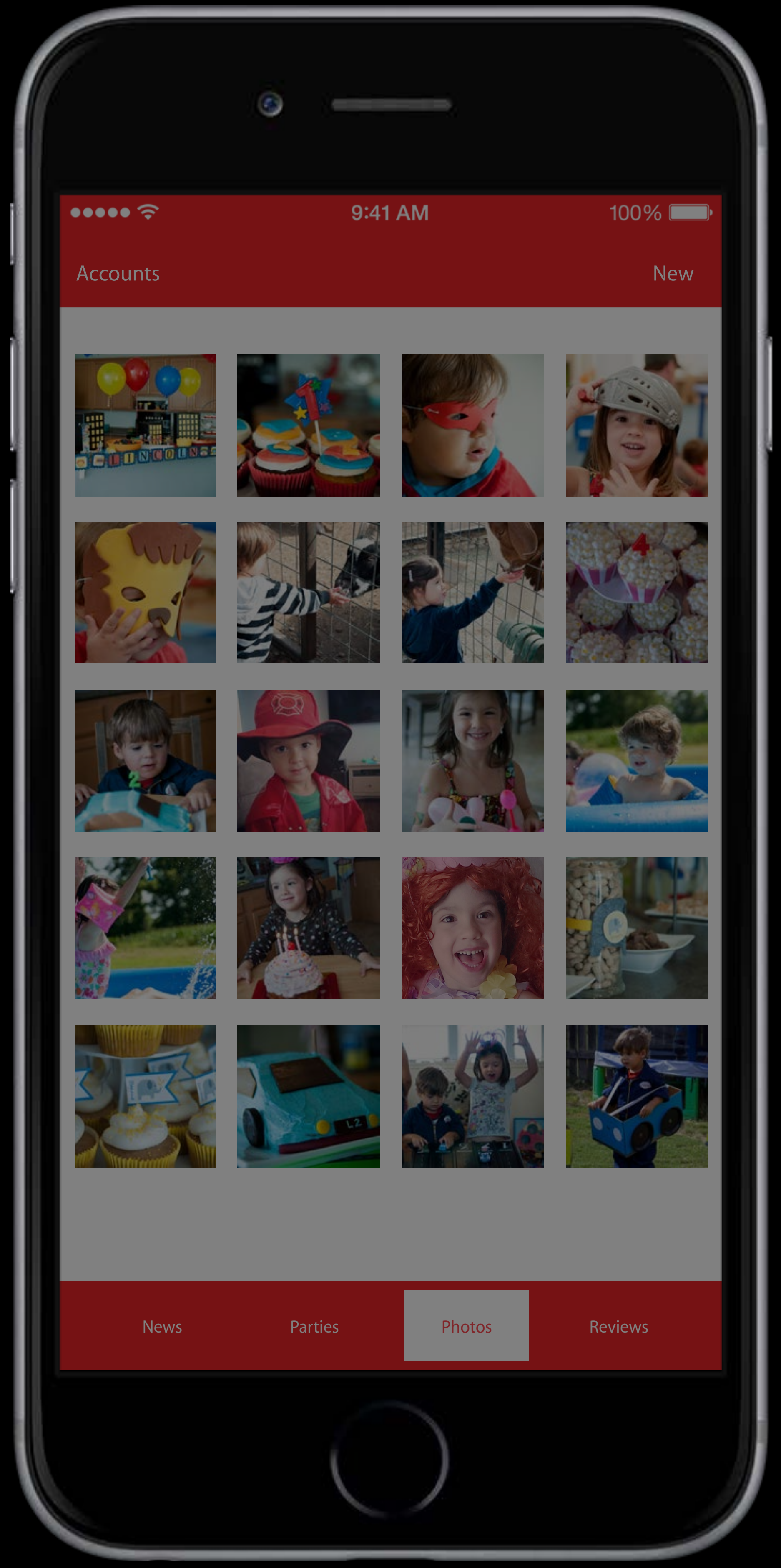
Photo



CKQueryOperation

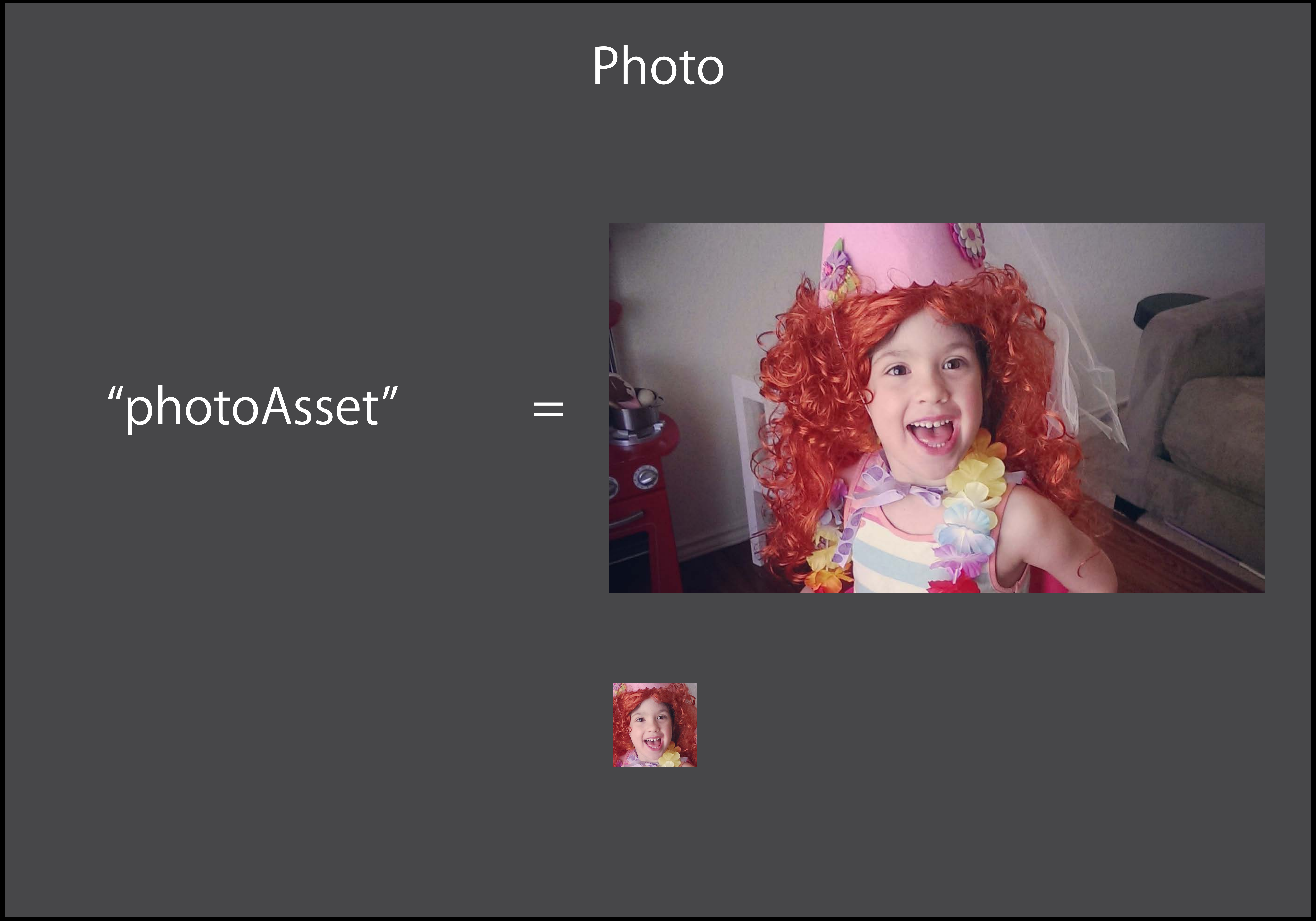
What kind of photo do we need?





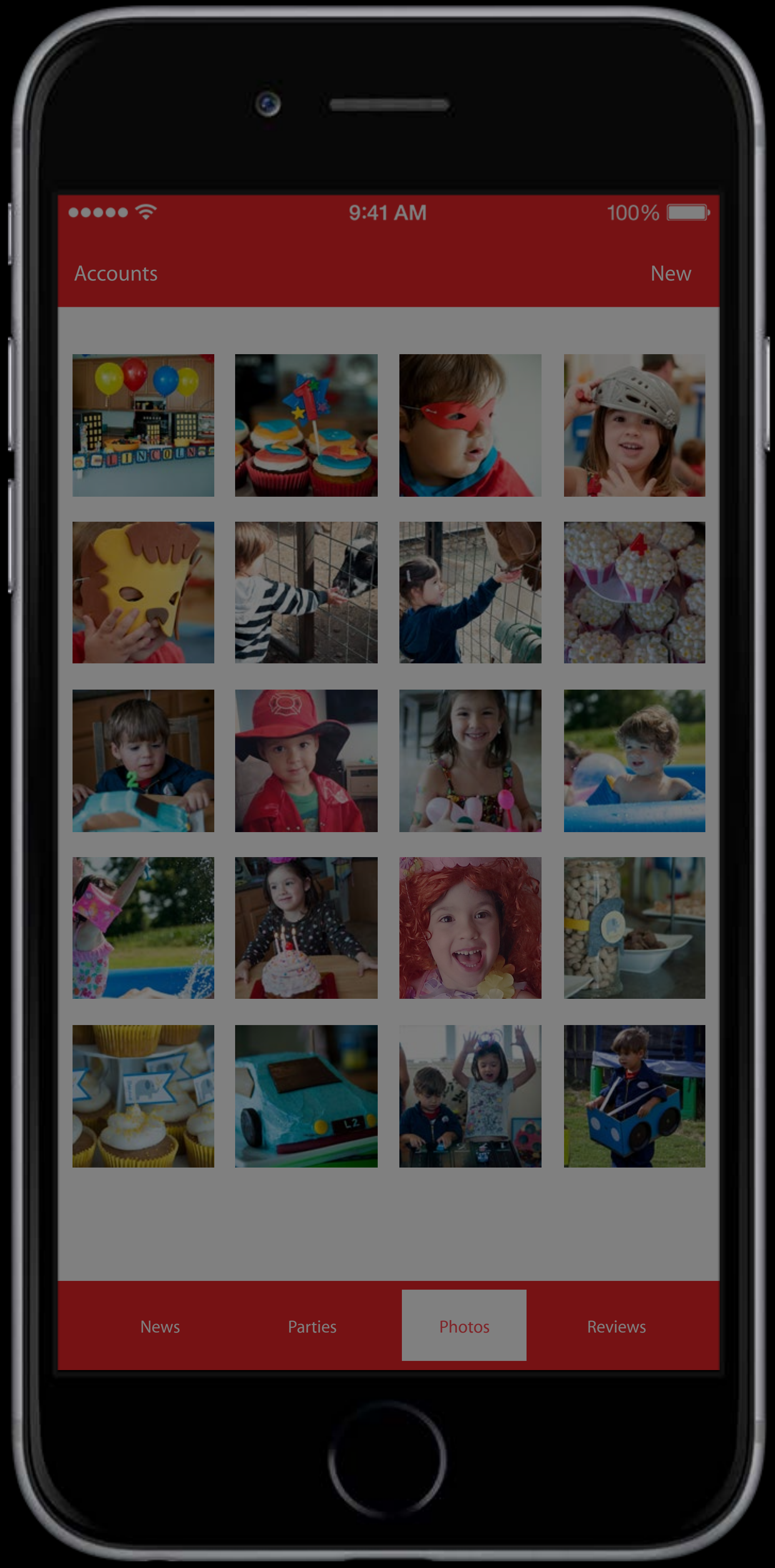
CKQueryOperation

What kind of photo do we need?



Photo

"photoAsset" =



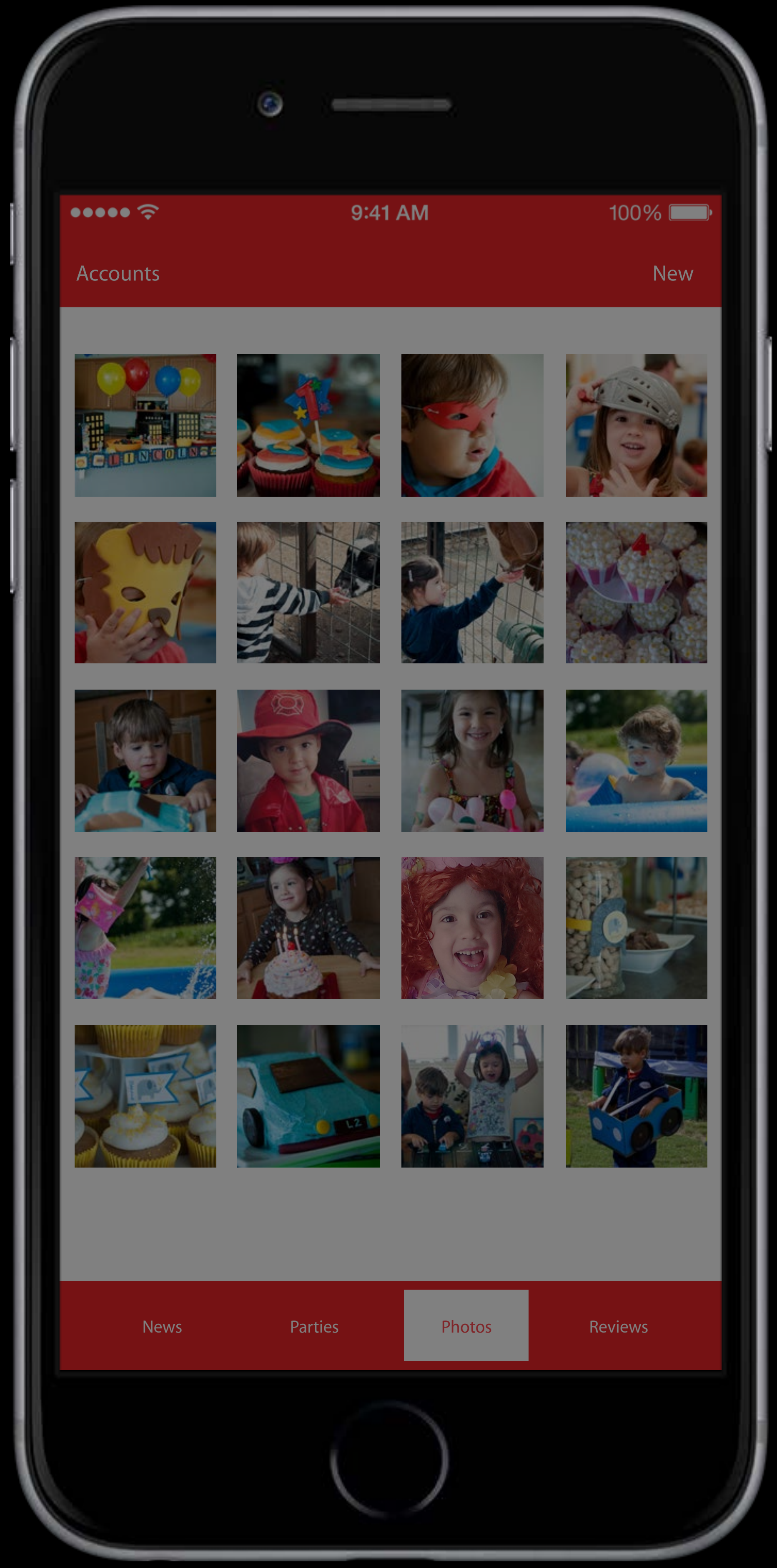
CKQueryOperation

What kind of photo do we need?

Photo

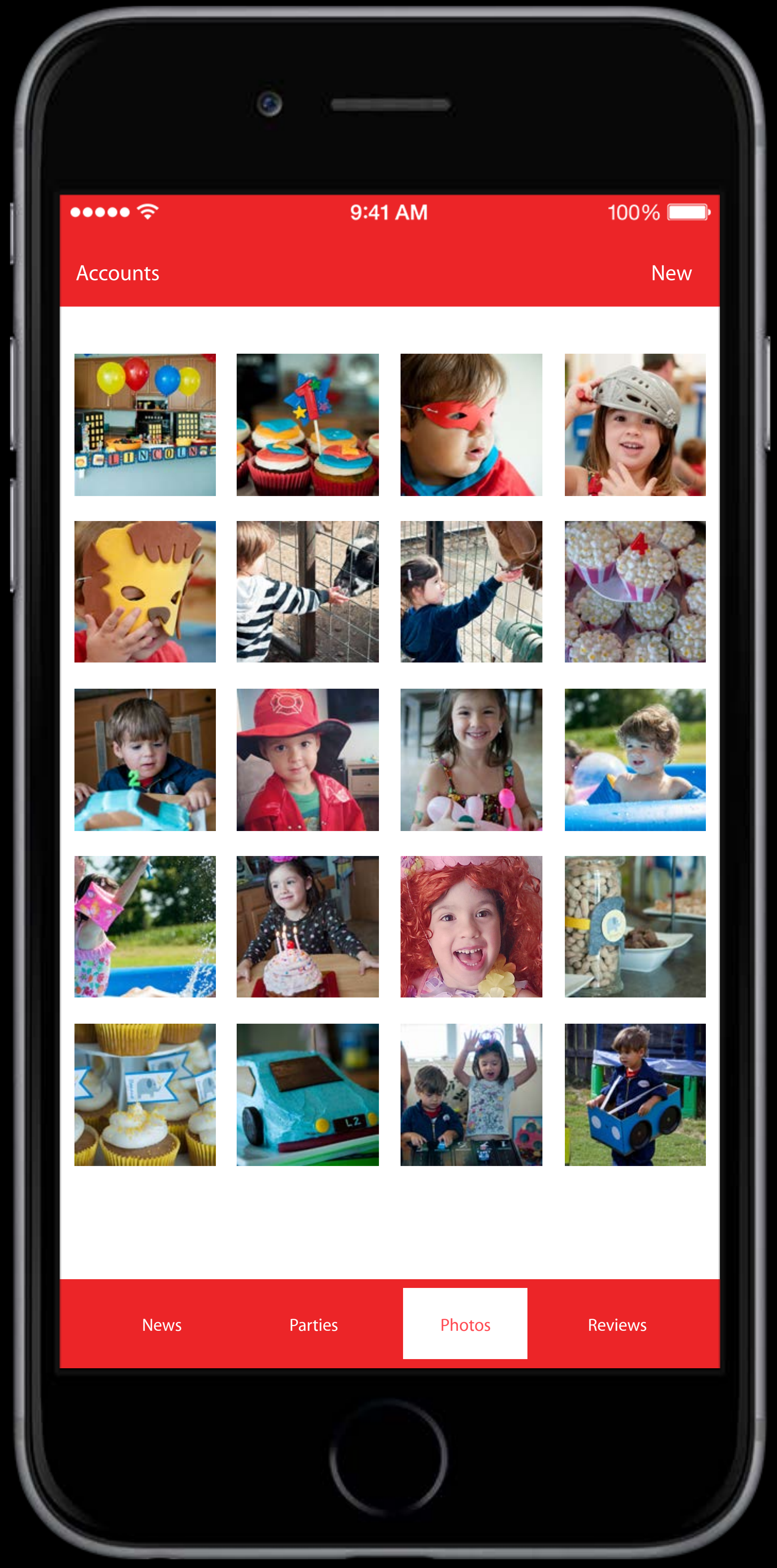
"photoAsset" =

"photoThumbnail" =



CKQueryOperation

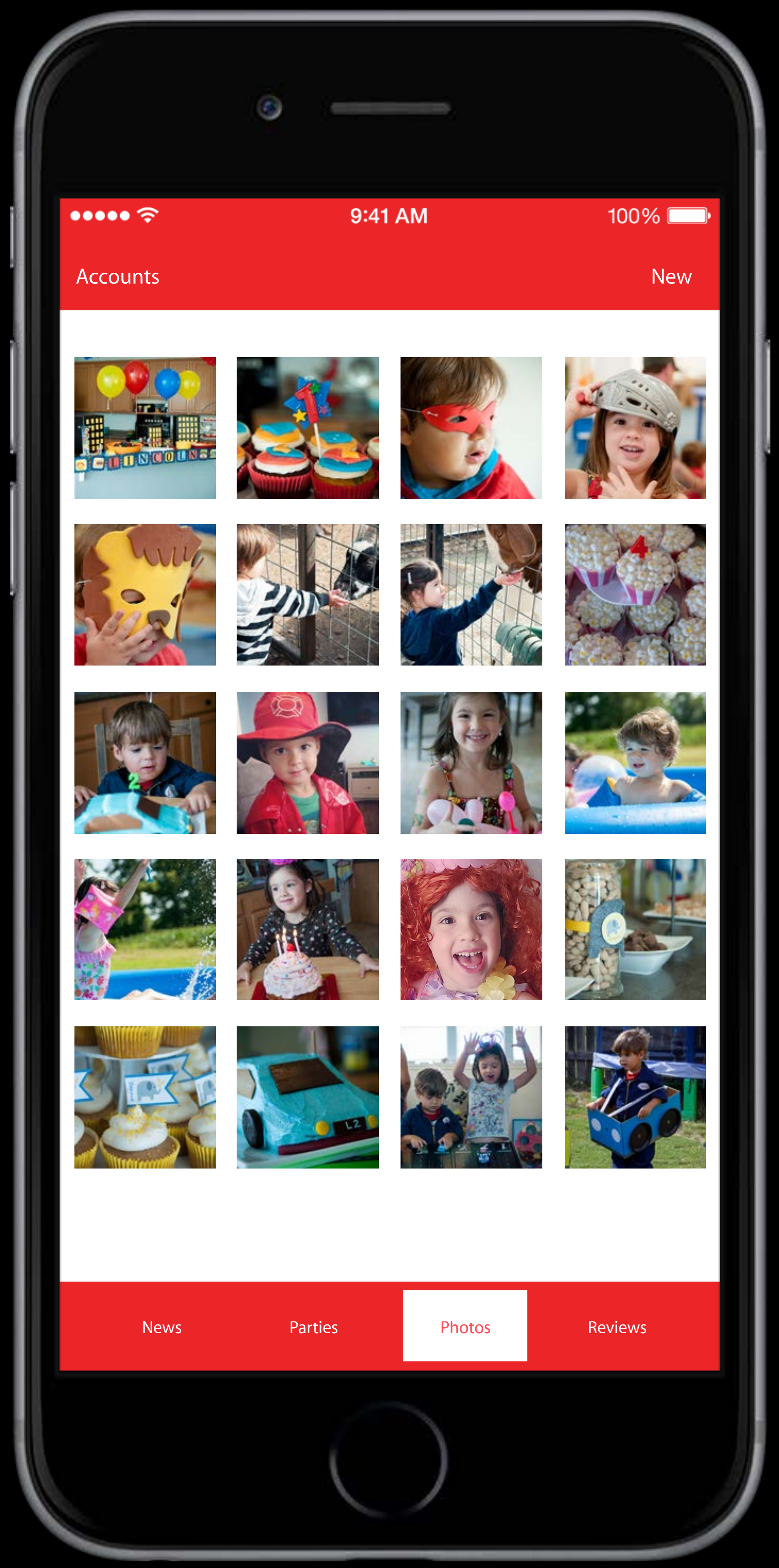
What kind of photo do we need?



CKQueryOperation

What kind of photo do we need?

```
let queryOperation = CKQueryOperation…
queryOperation.desiredKeys = ["photoThumbnail"]
```

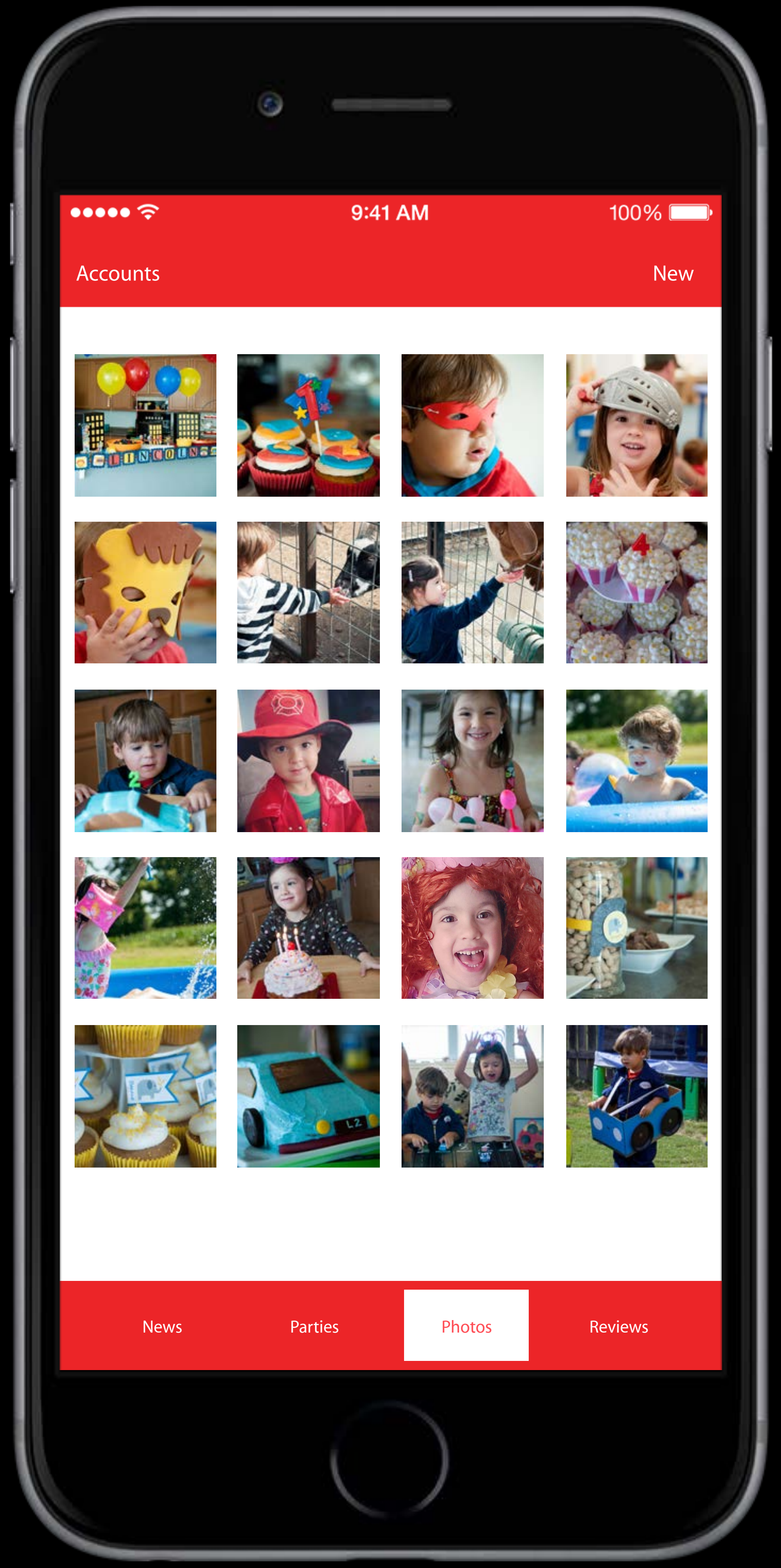


CKQueryOperation

What kind of photo do we need?

```
let queryOperation = CKQueryOperation…
queryOperation.desiredKeys = ["photoThumbnail"]
```

Also available on:



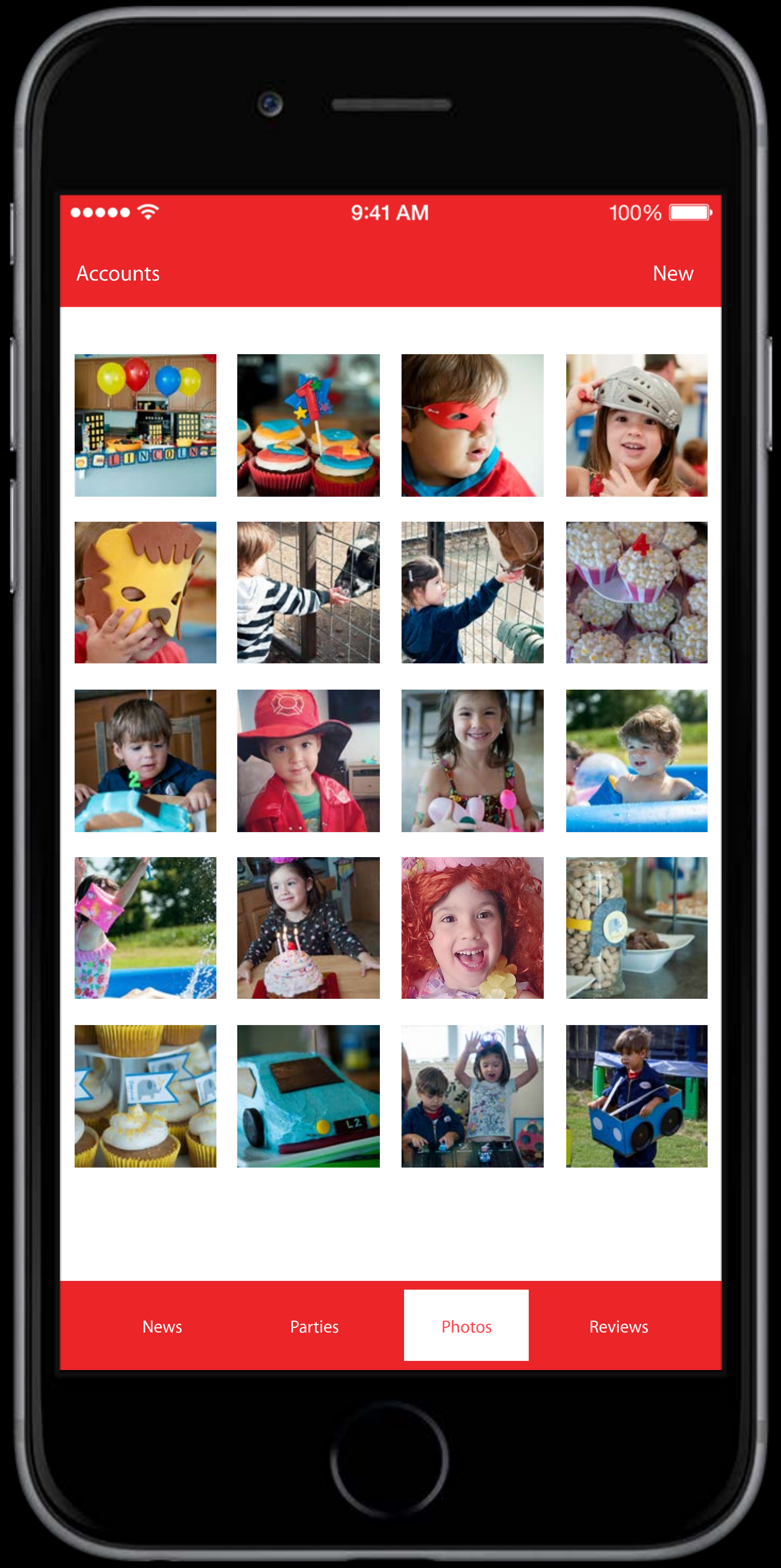
CKQueryOperation

What kind of photo do we need?

```
let queryOperation = CKQueryOperation…
queryOperation.desiredKeys = ["photoThumbnail"]
```

Also available on:

- `CKFetchRecordsOperation`



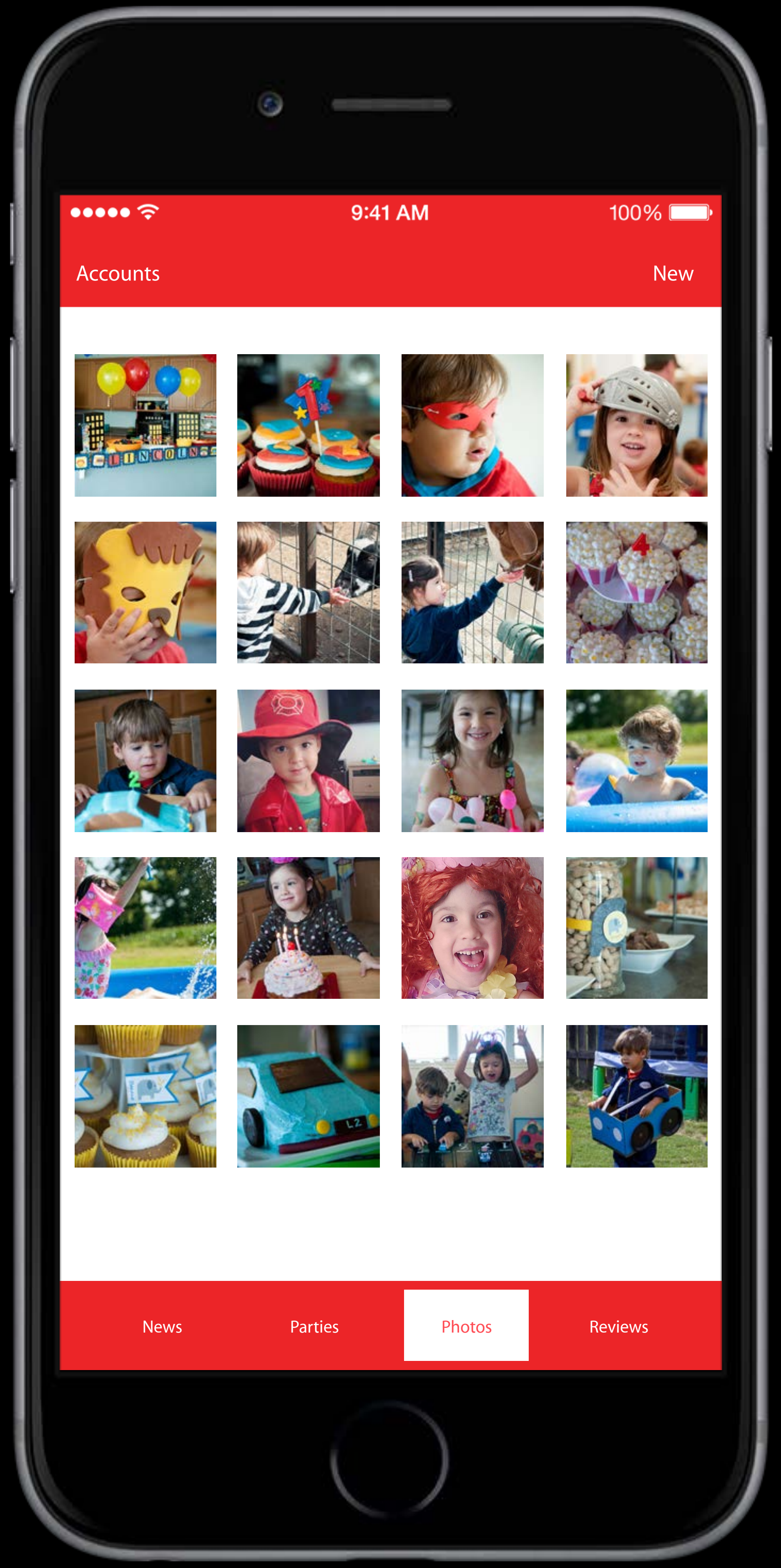
CKQueryOperation

What kind of photo do we need?

```
let queryOperation = CKQueryOperation…
queryOperation.desiredKeys = ["photoThumbnail"]
```

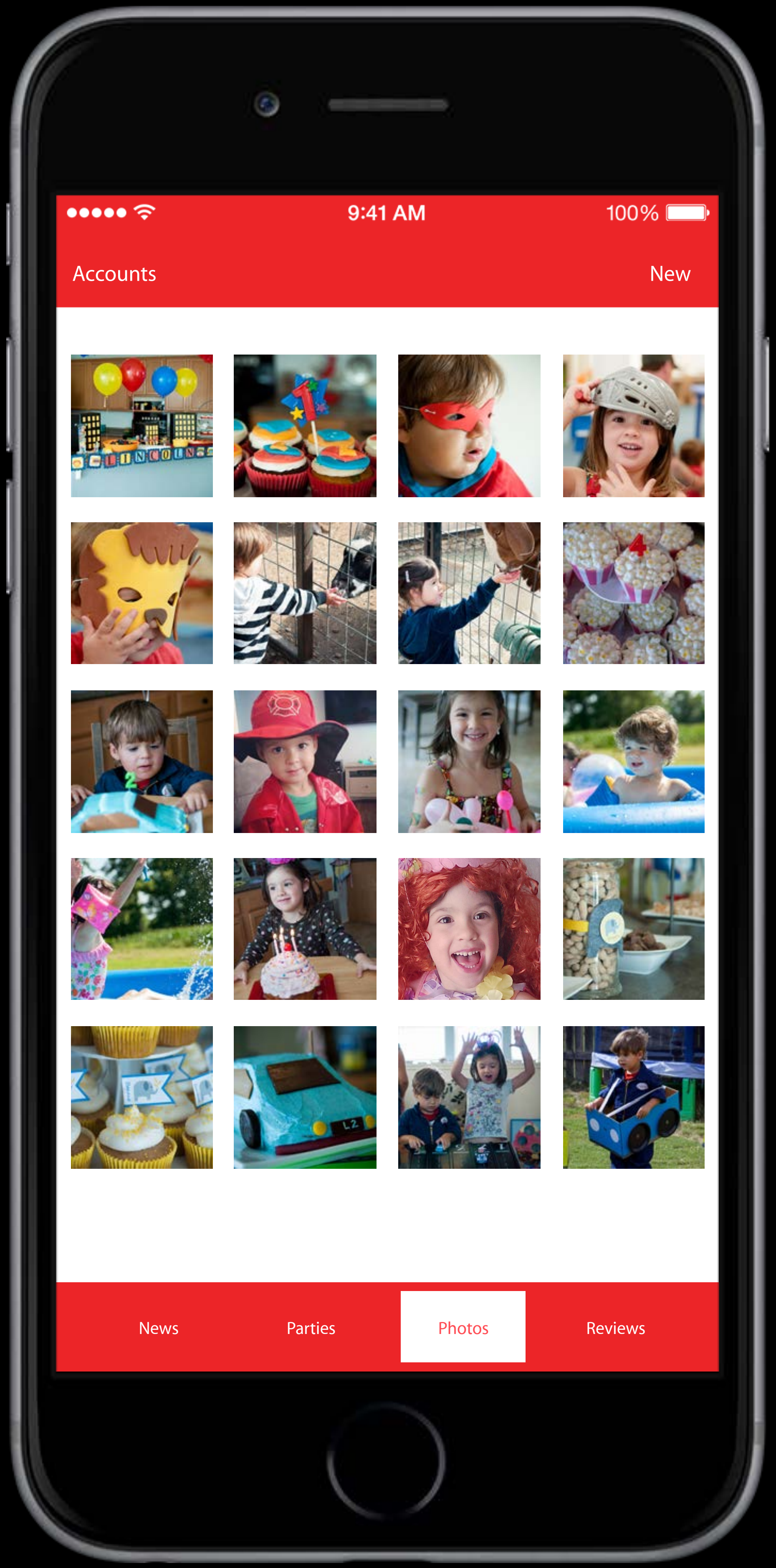
Also available on:

- `CKFetchRecordsOperation`
- `CKFetchRecordChangesOperation`



CKQueryOperation

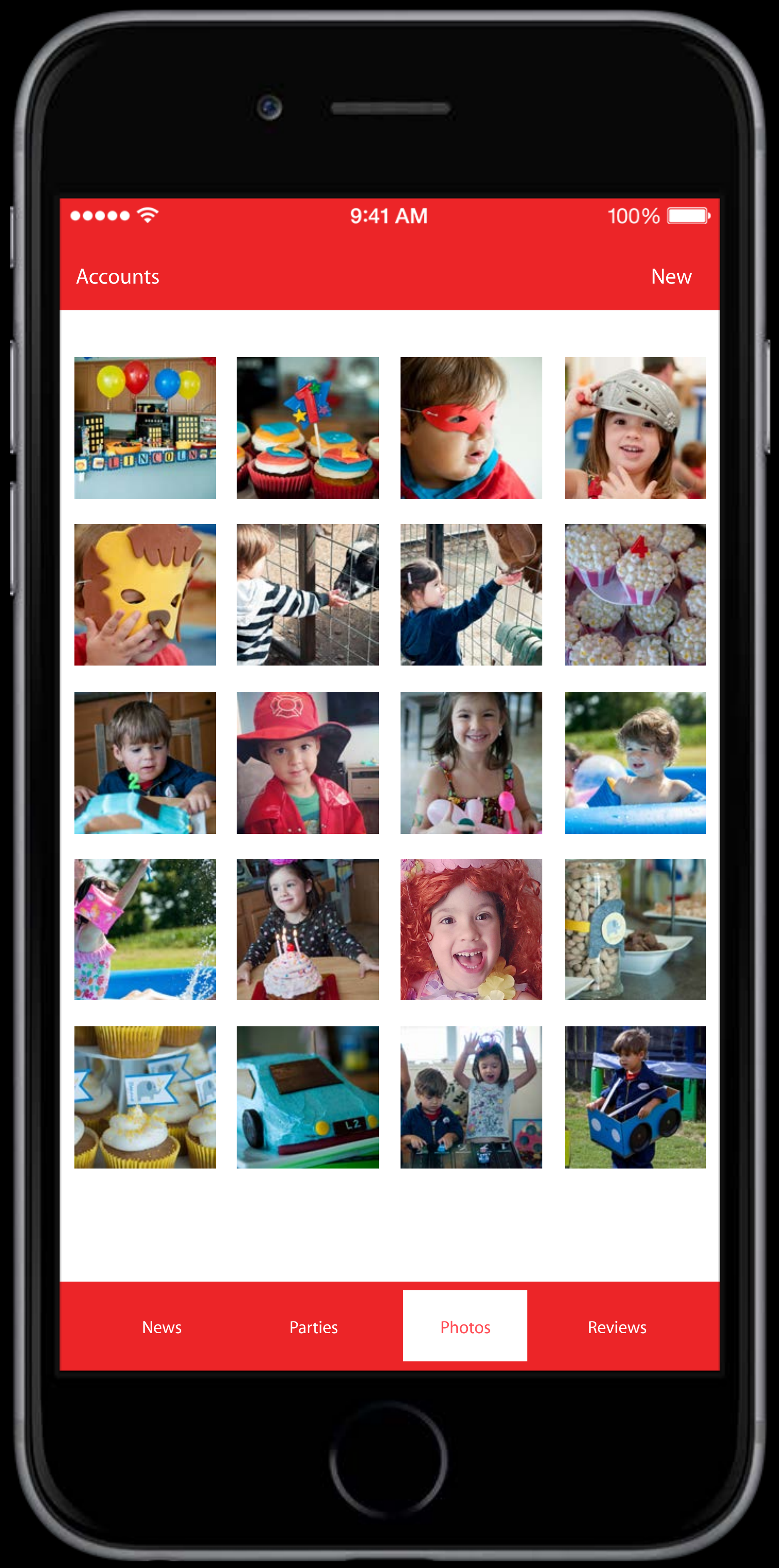
Fetching in order



CKQueryOperation

Fetching in order

```swift
let byCreation = NSSortDescriptor(key: "creationDate",
                                  ascending: false)
query.sortDescriptors = [byCreation]
```

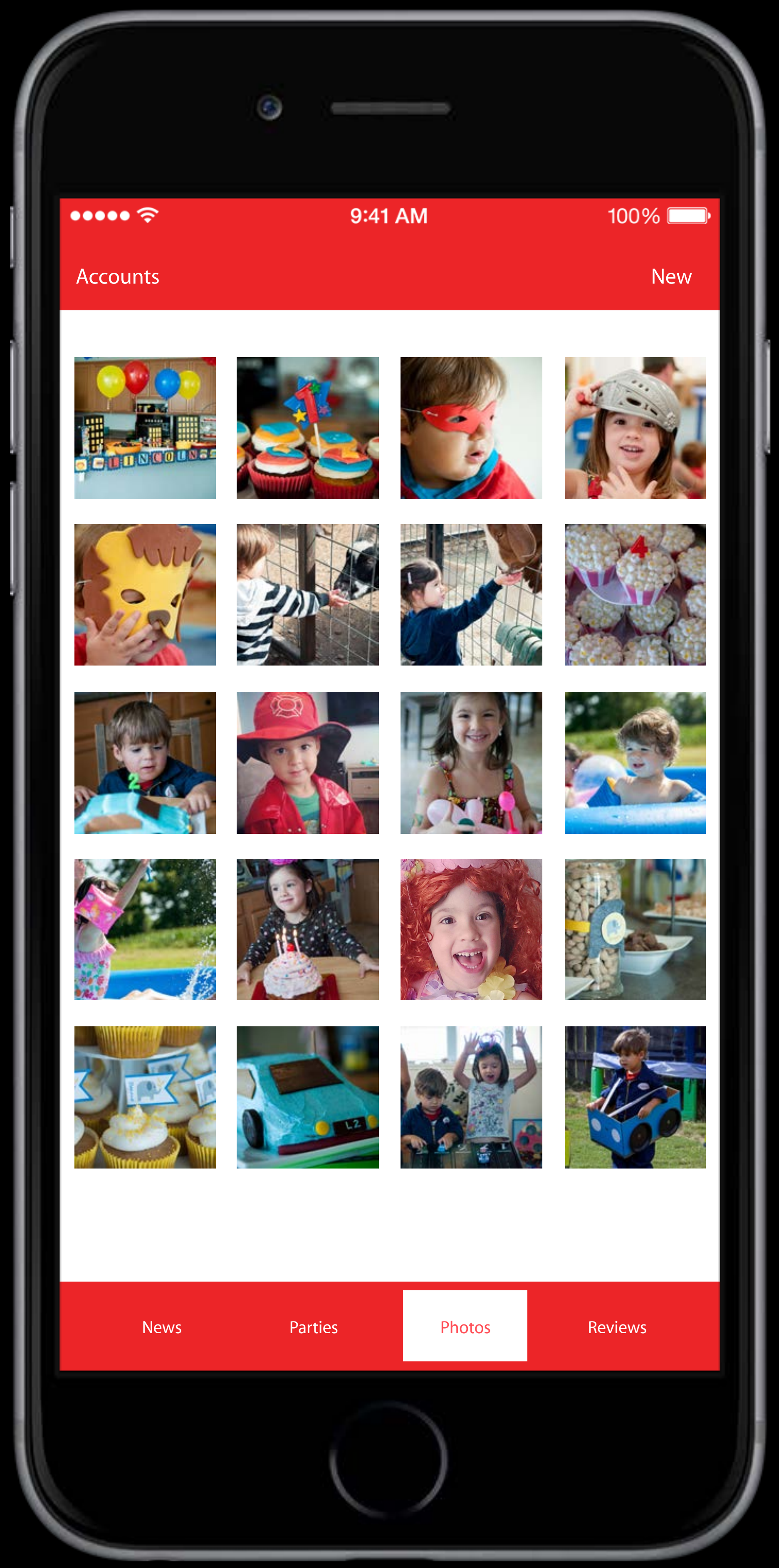


CKQueryOperation

Fetching in order

```
let byCreation = NSSortDescriptor(key: "creationDate",
                                  ascending: false)
query.sortDescriptors = [byCreation]
```

System field sort



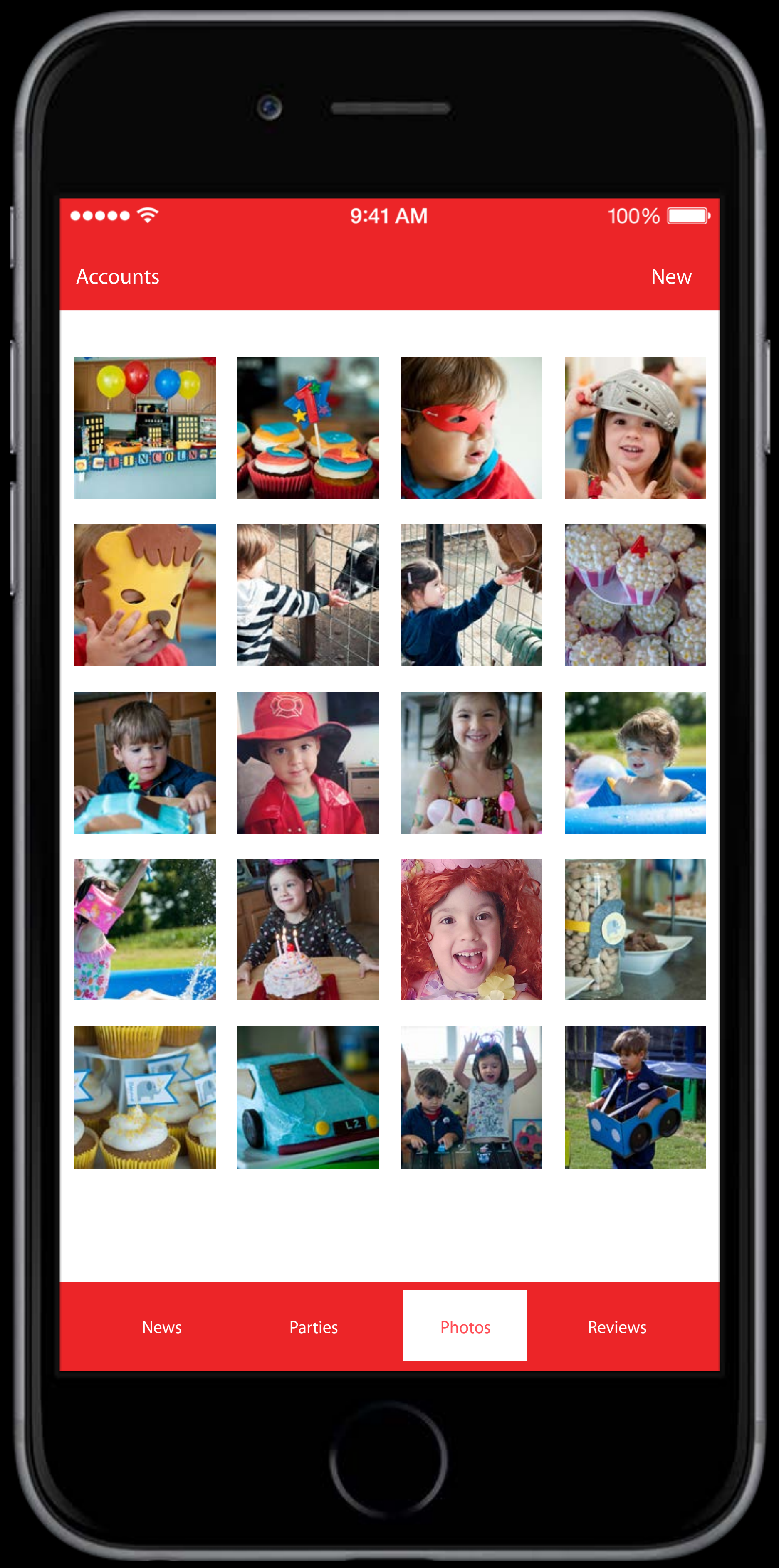
CKQueryOperation

Fetching in order

```
let byCreation = NSSortDescriptor(key: "creationDate",
                                  ascending: false)
query.sortDescriptors = [byCreation]
```

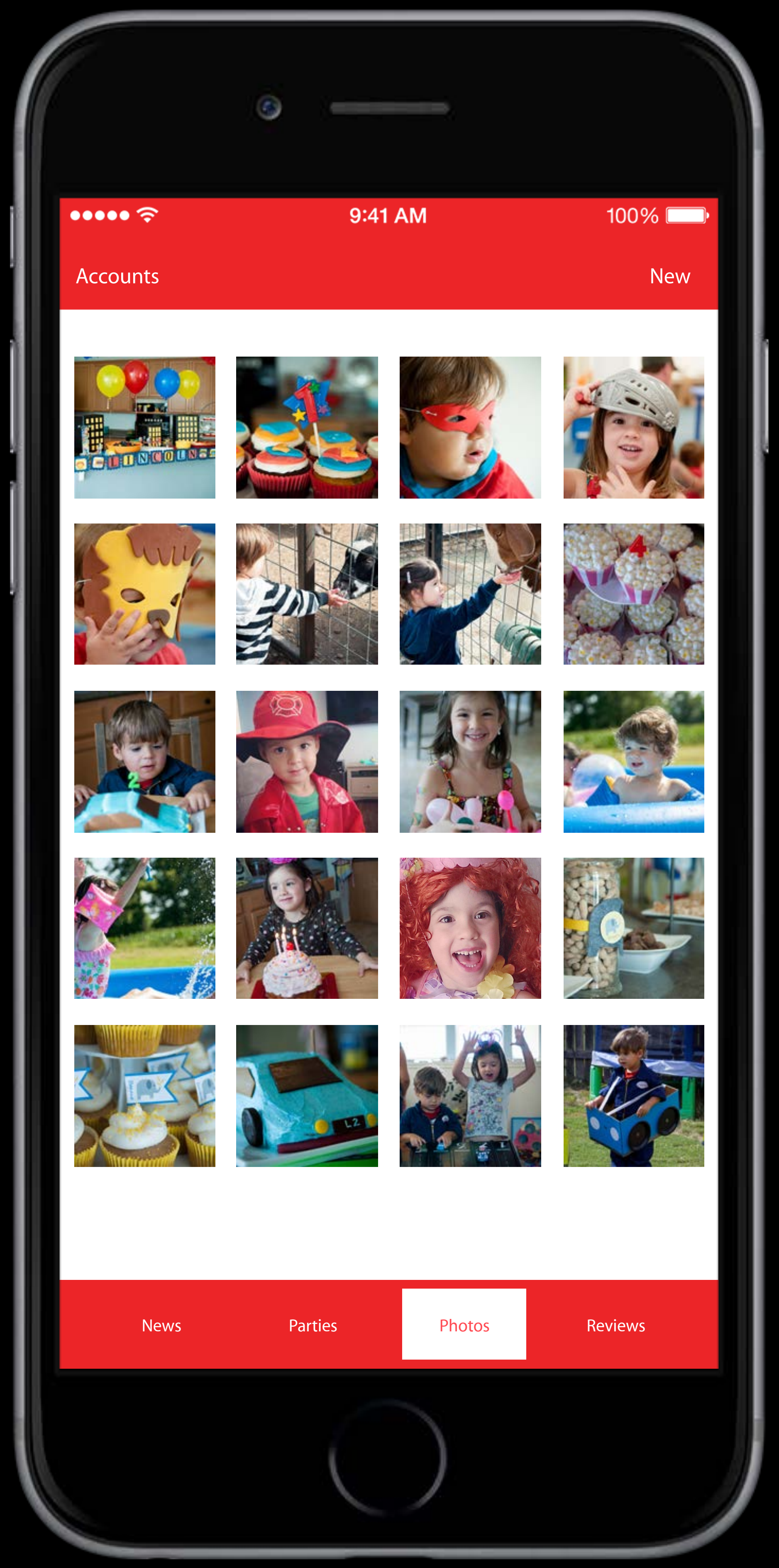
System field sort

- Configure via iCloud Dashboard



CKQueryOperation

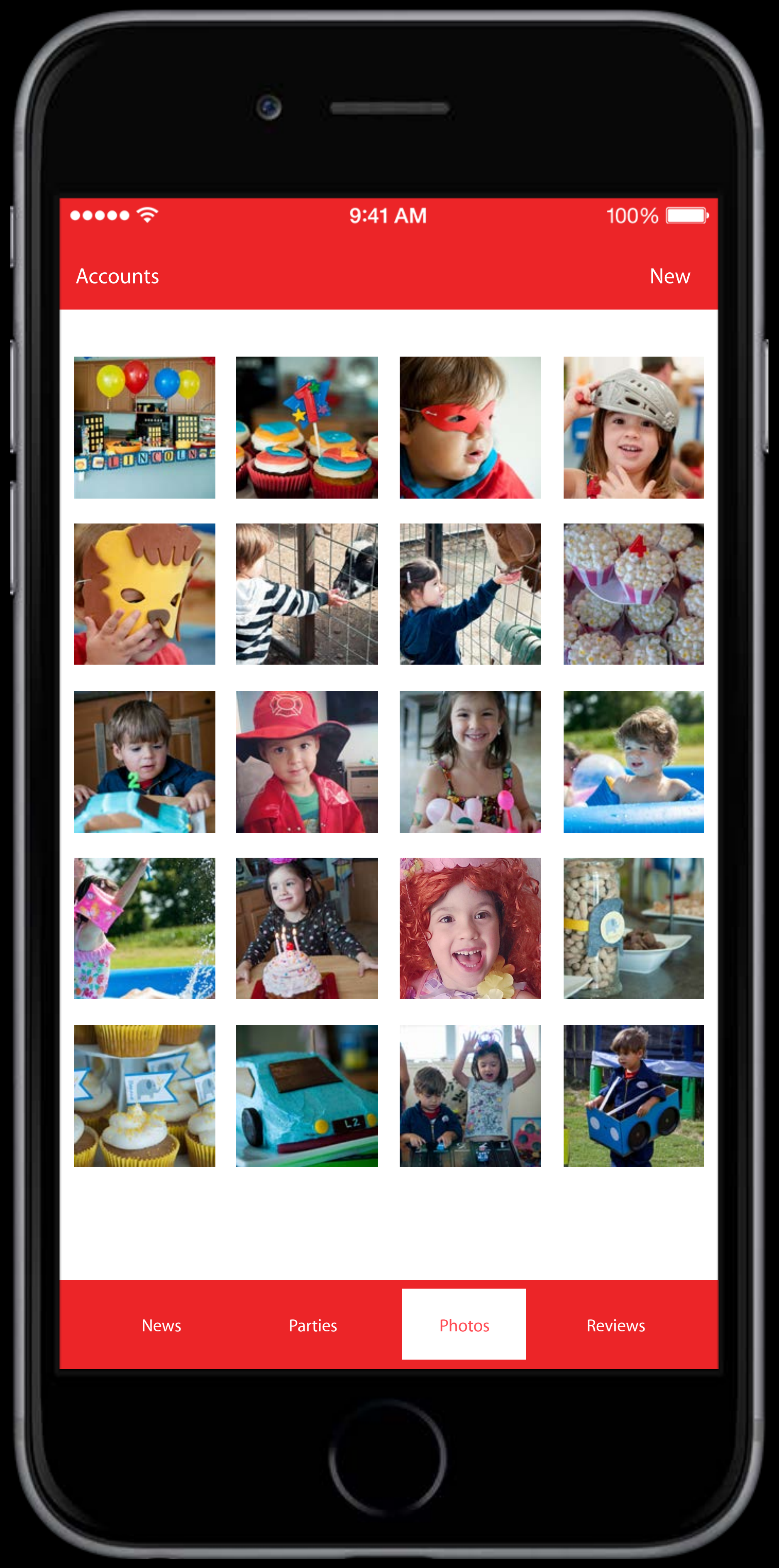
What about the rest?



CKQueryOperation

What about the rest?

Pagination

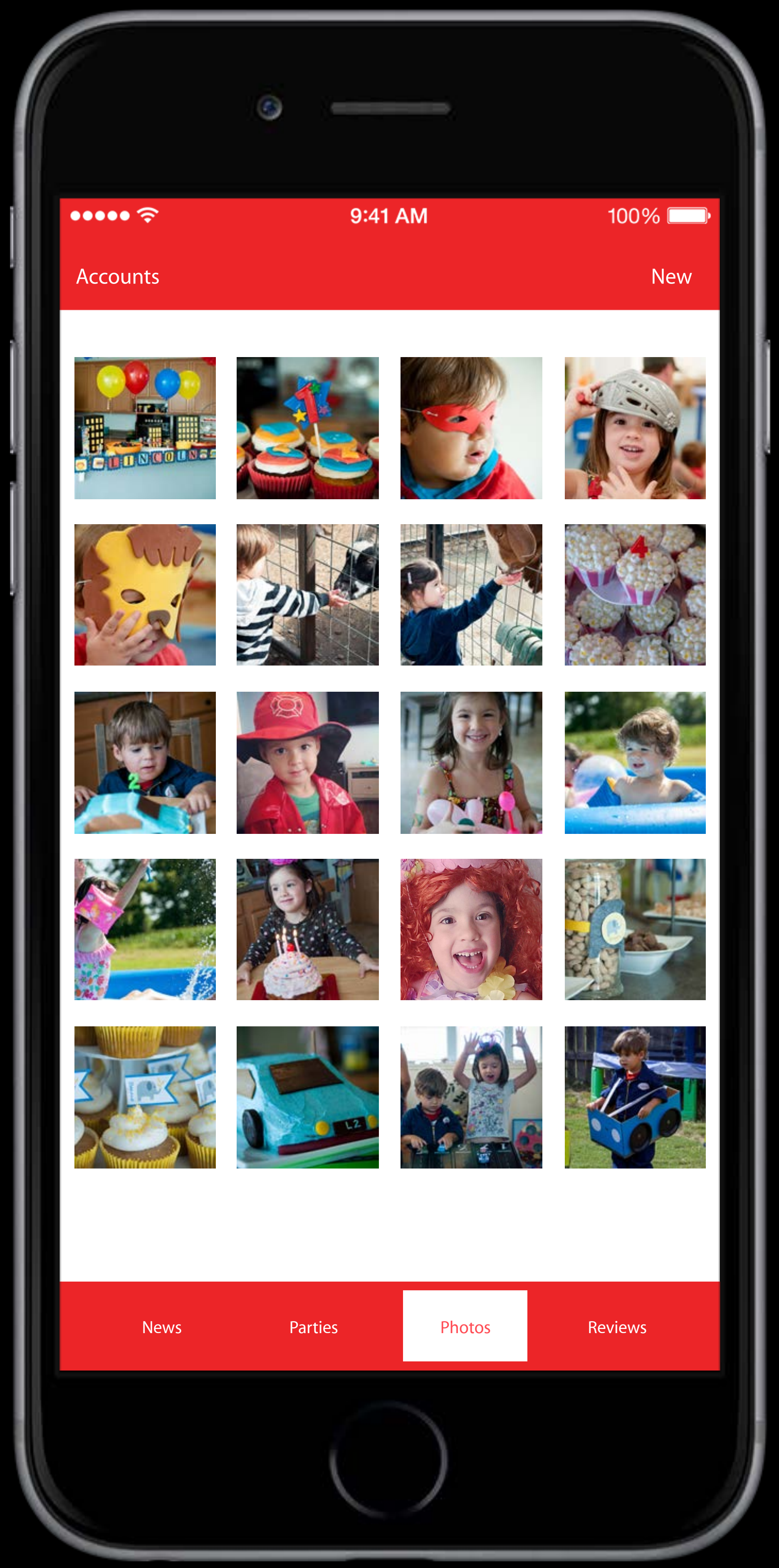


CKQueryOperation

What about the rest?

Pagination

```
var queryCompletionBlock:
((CKQueryCursor?, NSError?) -> Void)?
```



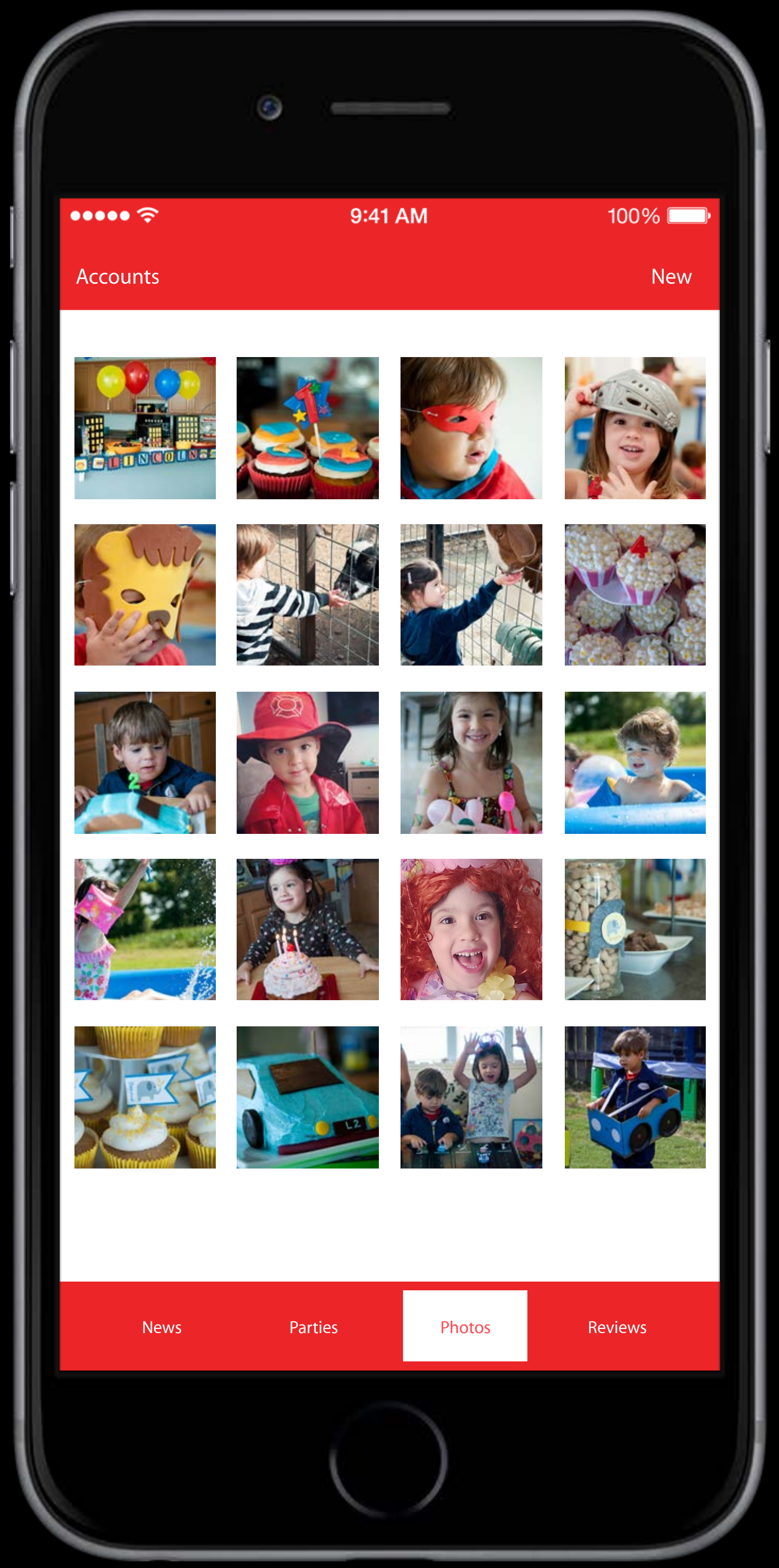
CKQueryOperation

What about the rest?

Pagination

```
var queryCompletionBlock:
((CKQueryCursor?, NSError?) -> Void)?
```

```
convenience init(cursor: CKQueryCursor)
```



Maintaining a Local Cache

Little Data, Many Clients

Little Data, Many Clients

Personal notes for parties

Little Data, Many Clients

Personal notes for parties

- Private database

Little Data, Many Clients

Personal notes for parties

- Private database
- Offline use

Little Data, Many Clients

Personal notes for parties

- Private database
- Offline use
- Small amount of data needed on all devices

Private Database Downloads

Private Database Downloads

```
let notesZone = CKRecordZone(zoneName: "NotesZone")
```

Private Database Downloads

```
let notesZone = CKRecordZone(zoneName: "NotesZone")
```

Two main ways:

Private Database Downloads

```
let notesZone = CKRecordZone(zoneName: "NotesZone")
```

Two main ways:

- `CKQueryOperation`

Private Database Downloads

```
let notesZone = CKRecordZone(zoneName: "NotesZone")
```

Two main ways:

- `CKQueryOperation`
- Delta downloads using `CKFetchRecordChangesOperation`

Private Database Downloads

```
let notesZone = CKRecordZone(zoneName: "NotesZone")
```

Two main ways:

- `CKQueryOperation`
- Delta downloads using `CKFetchRecordChangesOperation`
    - Custom zone with `CKRecordZoneCapabilityFetchChanges`

Storing Data Locally

App objects

Party	
PartyName	My Party
PartyDate	June 11, 2015
PartyNotes	<TextBlob>

Local Store

Storing Data Locally

App objects

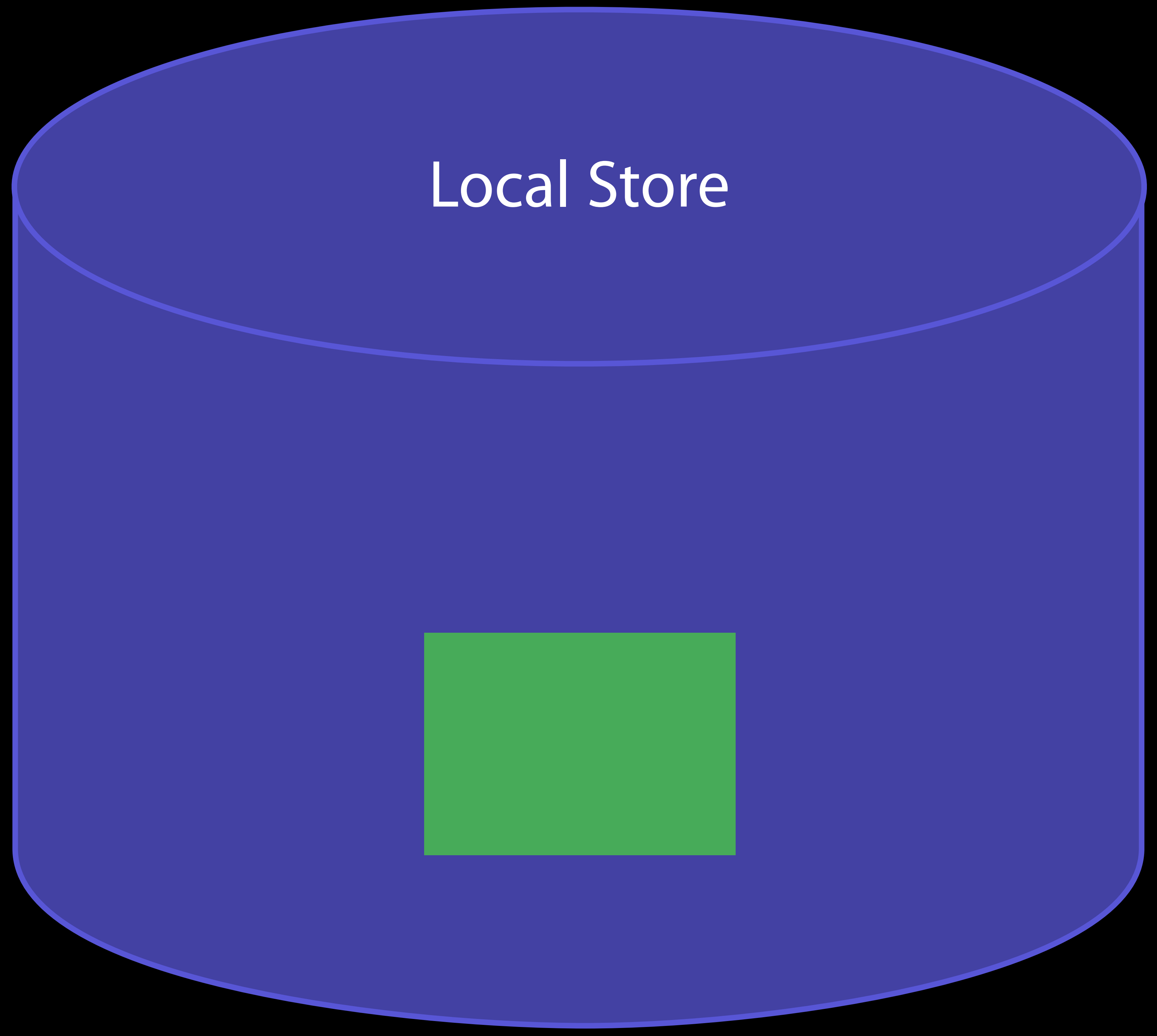
Party	
PartyName	My Party
PartyDate	June 11, 2015
PartyNotes	<TextBlob>

Local Store

Storing Data Locally

CKRecord

Party	
Record ID	FDFE37608477
ChangeTag	A
PartyName	My Party
PartyDate	June 11, 2015
PartyNotes	<TextBlob>



Storing Data Locally

CKRecord

Party	
Record ID	FDFE37608477
ChangeTag	A
PartyName	My Party
PartyDate	June 11, 2015
PartyNotes	<TextBlob>

Local Store

Storing Data Locally

CKRecord

Party	
Record ID	FDFE37608477
ChangeTag	A
PartyName	My Party
PartyDate	June 11, 2015
PartyNotes	<TextBlob>

Local Store

Storing Records Locally

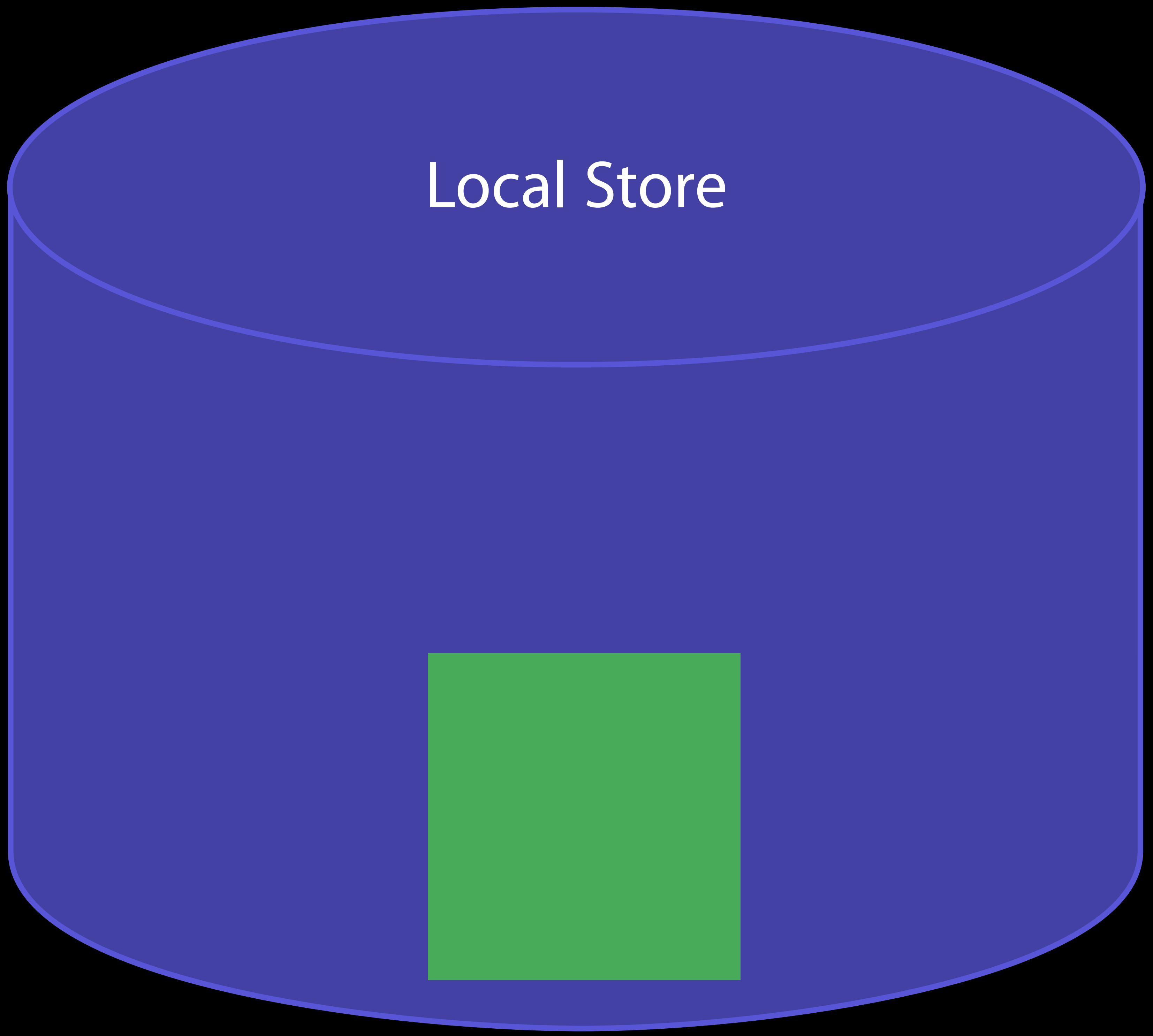
Archiving system fields

```
let record = …

let archivedData = NSMutableData()
let archiver = NSKeyedArchiver(forWritingWithMutableData: archivedData)
archiver.requiresSecureCoding = true
record.encodeSystemFieldsWithCoder(archiver)
archiver.finishEncoding()
```

Storing Records Locally

Archiving system fields

Party	
Record ID	FDFE37608477
ChangeTag	A
PartyName	My Party
PartyDate	June 11, 2015
PartyNotes	<TextBlob>

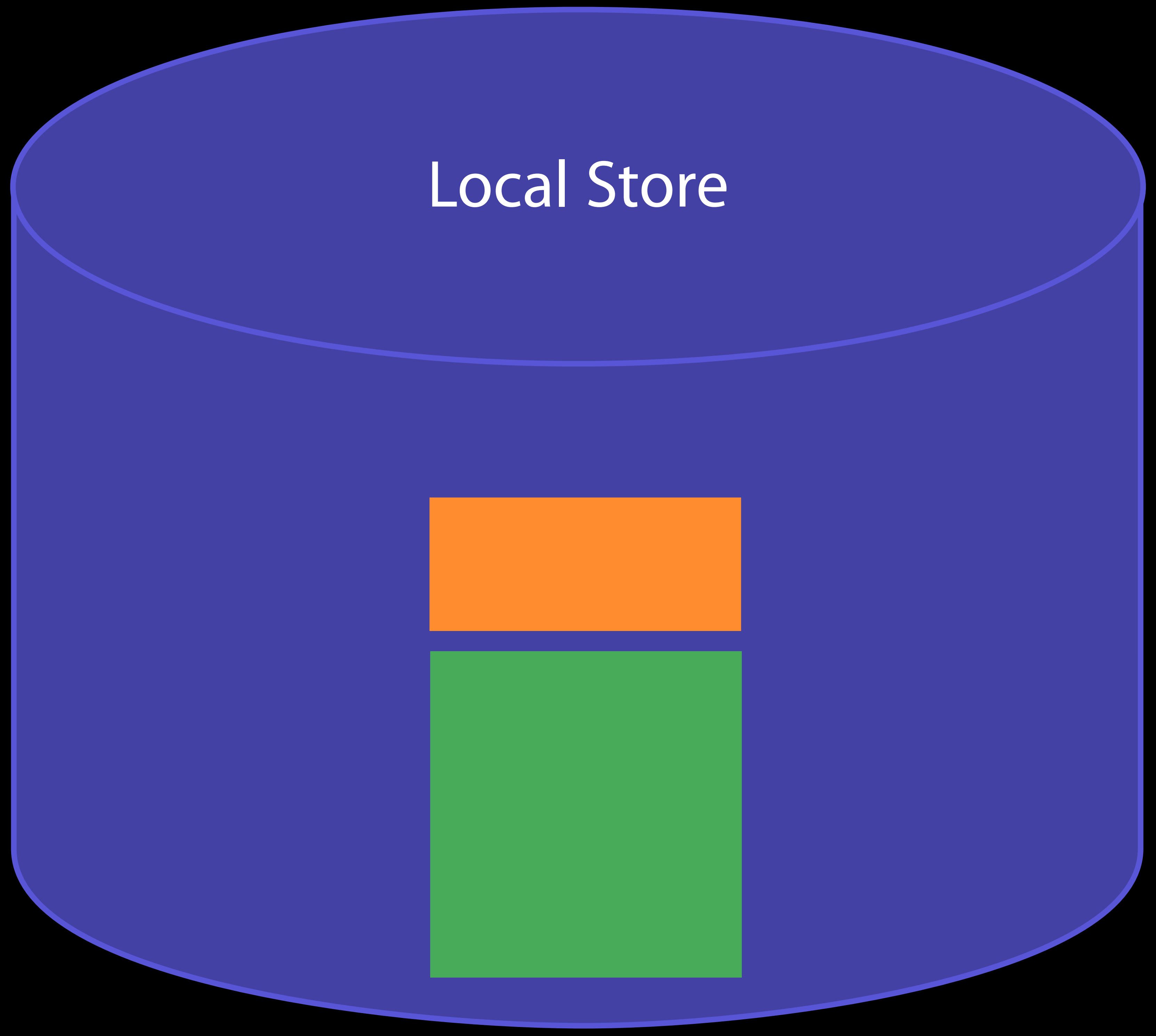


Local Store

Storing Records Locally

Archiving system fields

Party	
Record ID	FDFE37608477
ChangeTag	A
PartyName	My Party
PartyDate	June 11, 2015
PartyNotes	<TextBlob>

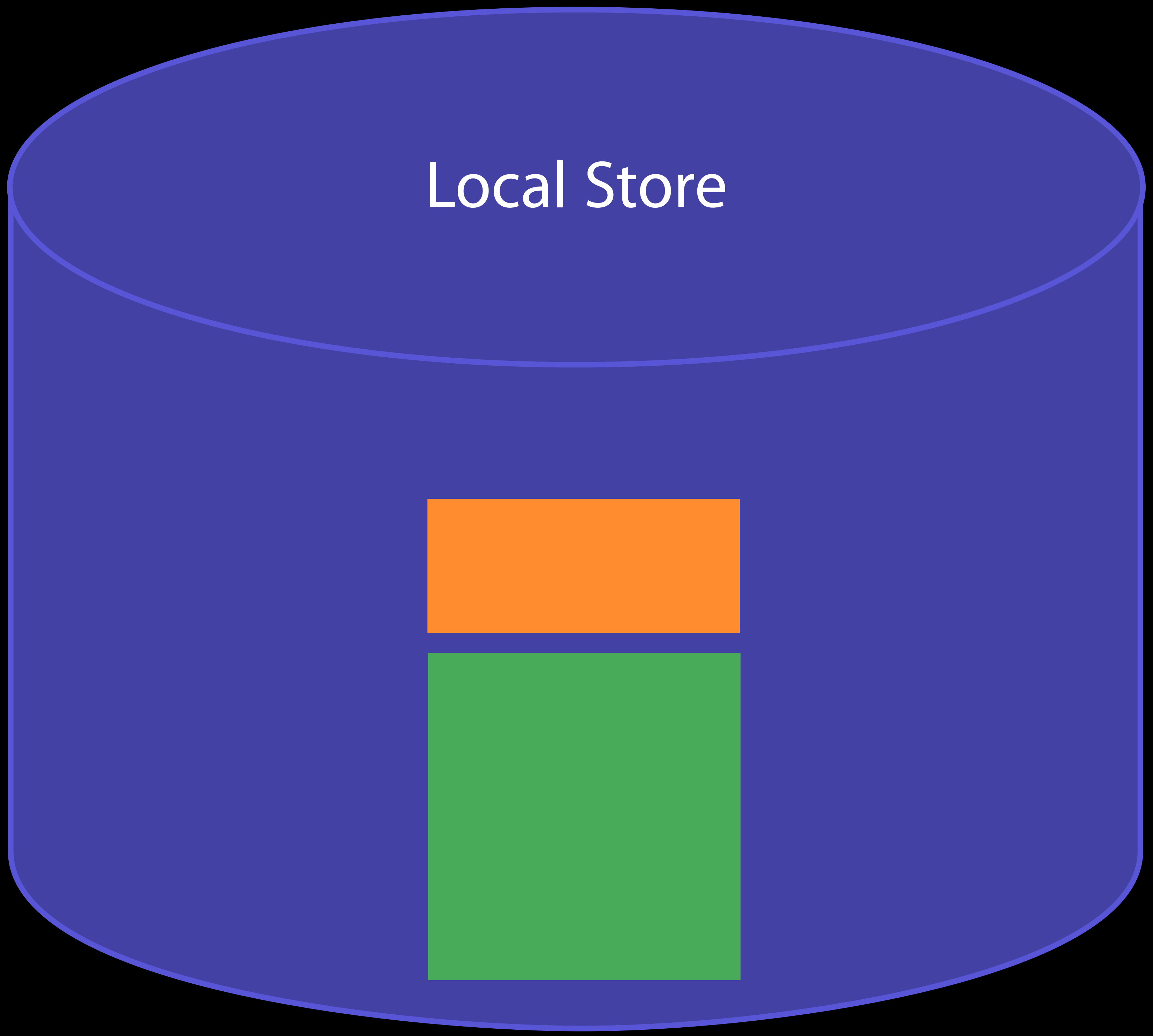


Storing Records Locally

Archiving system fields

✓

Party	
Record ID	FDFE37608477
ChangeTag	A
PartyName	My Party
PartyDate	June 11, 2015
PartyNotes	<TextBlob>



Storing Records Locally

Unarchiving system fields

```
let record = ...

let archivedData = NSMutableData()
let archiver = NSKeyedArchiver(forWritingWithMutableData: archivedData)
archiver.requiresSecureCoding = true
record.encodeSystemFieldsWithCoder(archiver)
archiver.finishEncoding()
```

Storing Records Locally

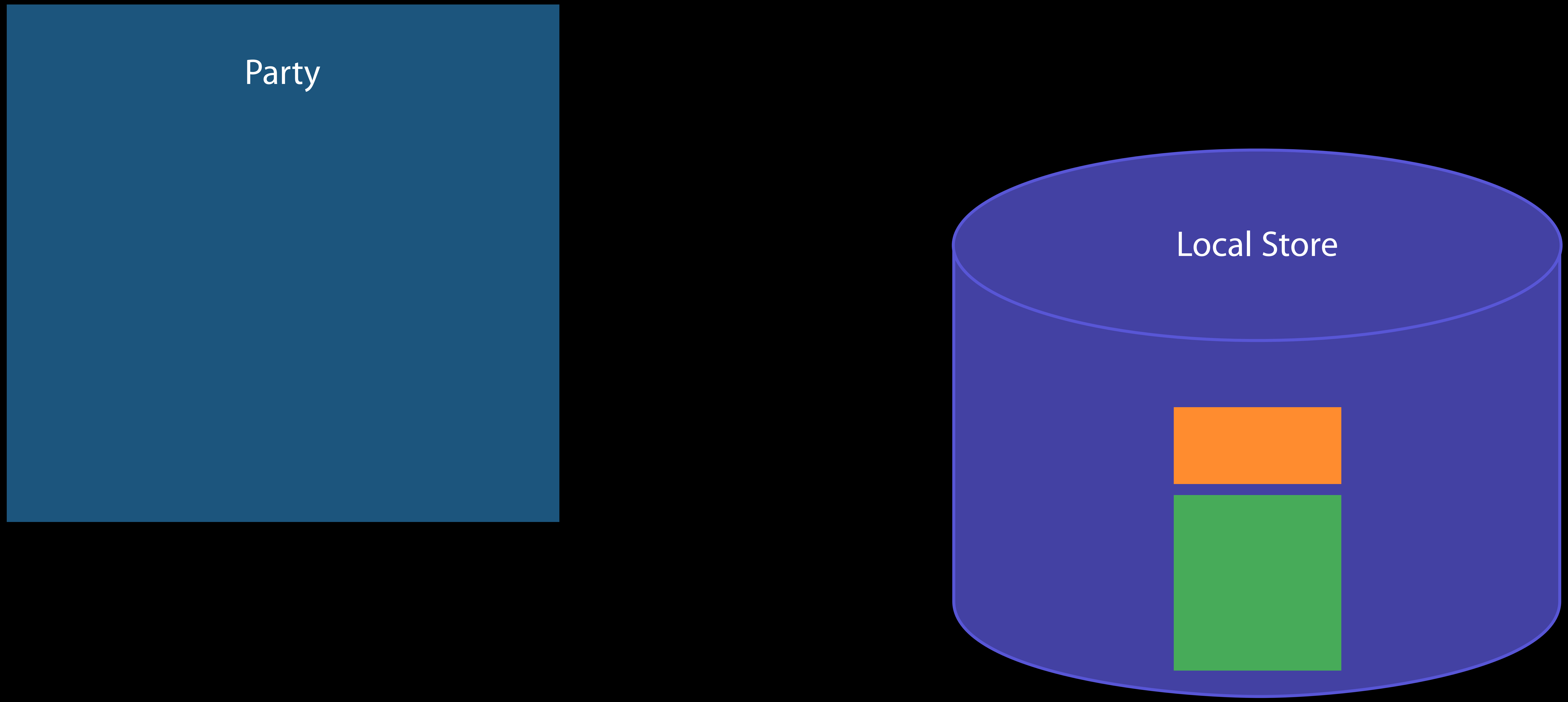
Unarchiving system fields

```
let record = ...

let archivedData = NSMutableData()
let archiver = NSKeyedArchiver(forWritingWithMutableData: archivedData)
archiver.requiresSecureCoding = true
record.encodeSystemFieldsWithCoder(archiver)
archiver.finishEncoding()
```

```
let unarchiver = NSKeyedUnarchiver(forReadingWithData: archivedData)
unarchiver.requiresSecureCoding = true
let unarchivedRecord = CKRecord(coder: unarchiver)
```

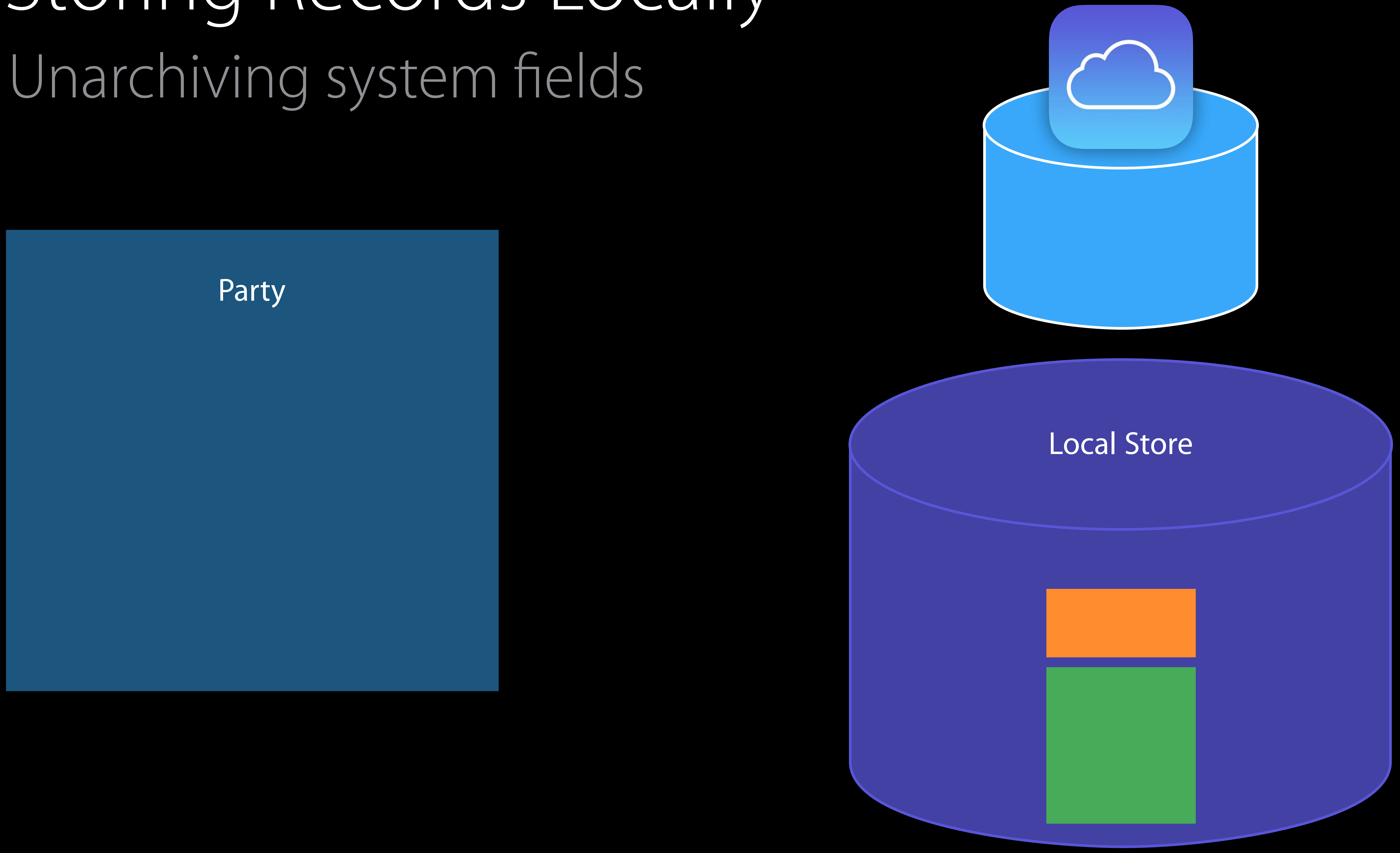
Storing Records Locally

Unarchiving system fields



Storing Records Locally

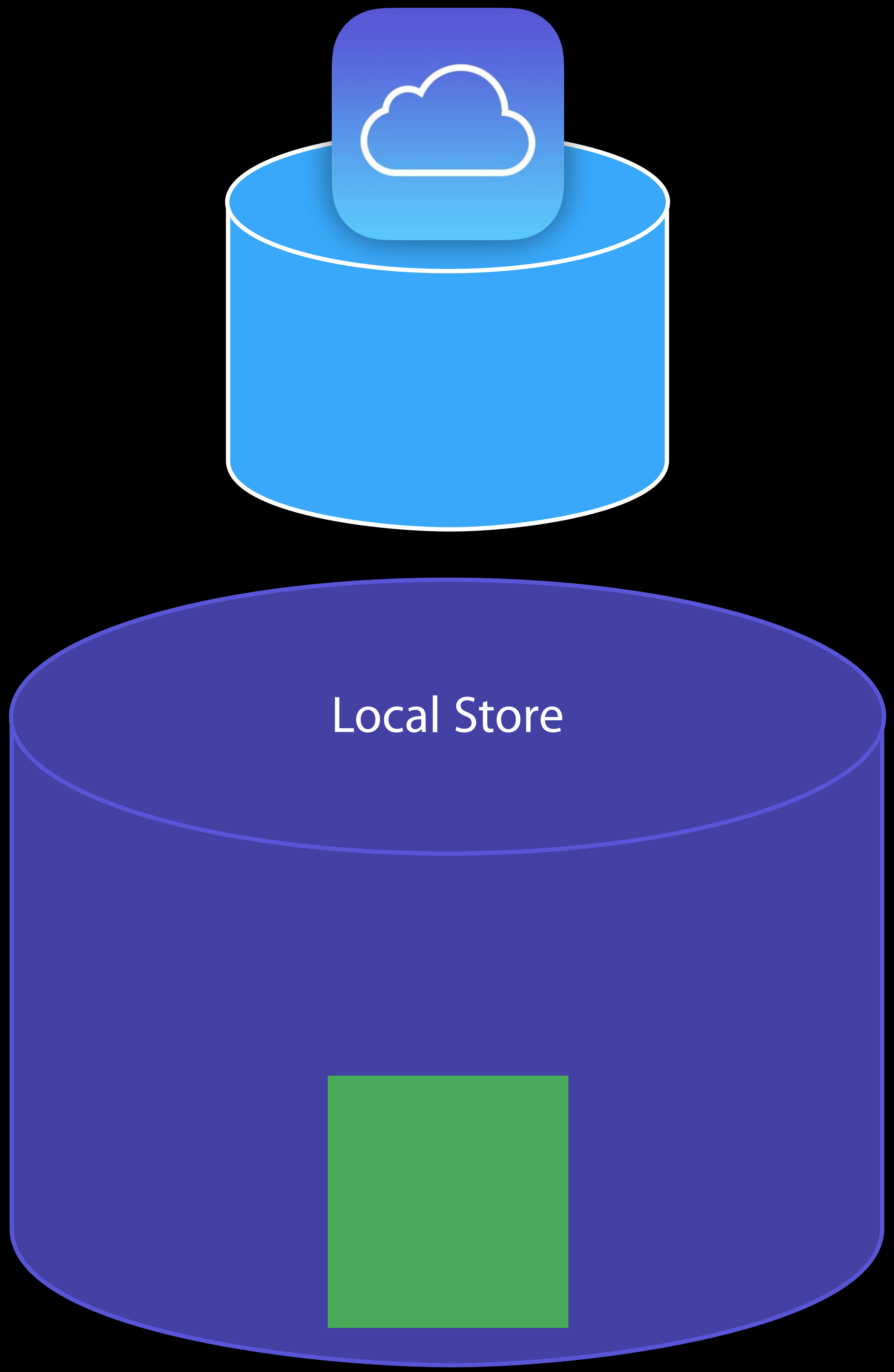
Unarchiving system fields



Storing Records Locally

Unarchiving system fields

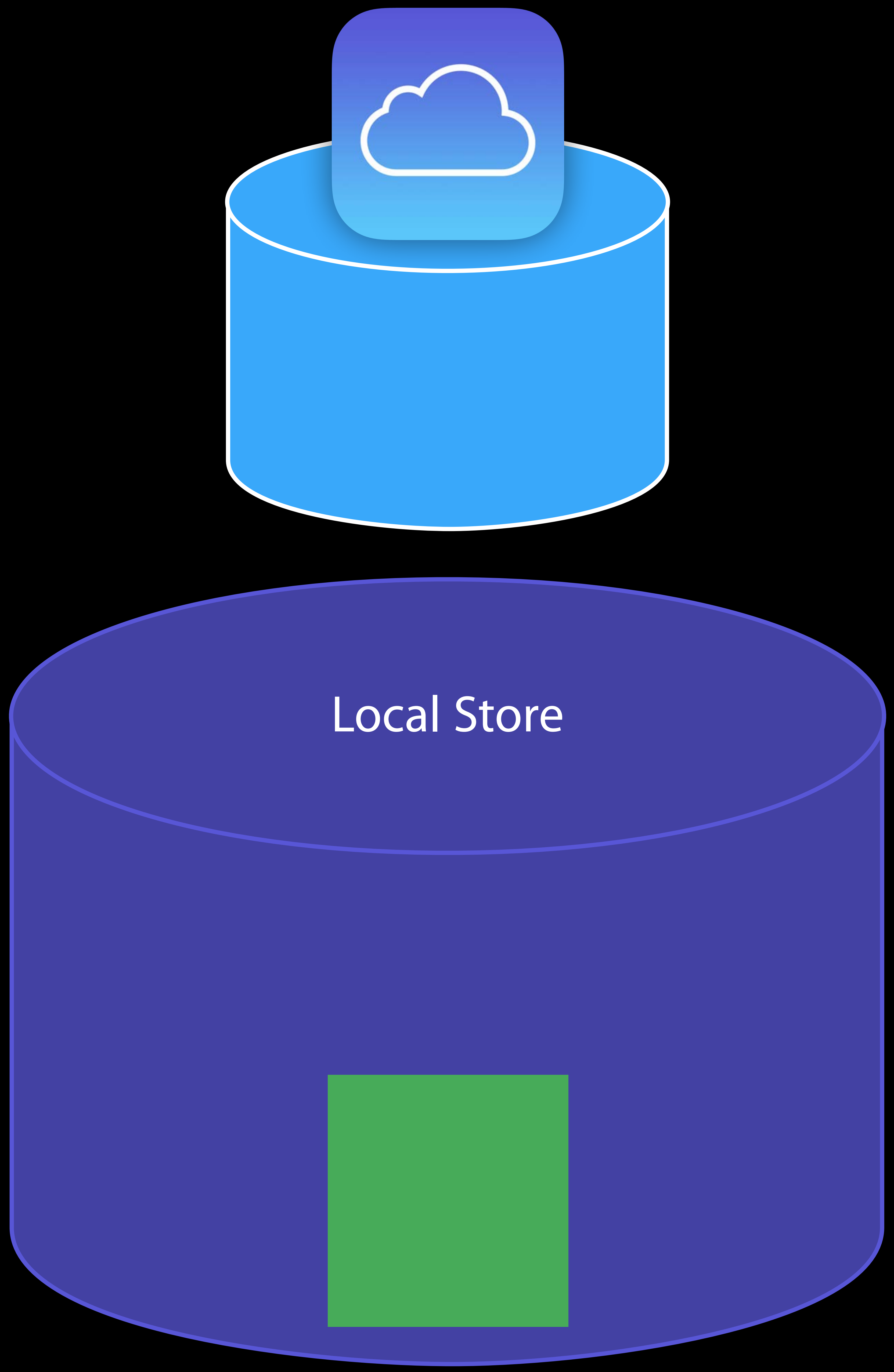
Party	
Record ID	FDFE37608477
ChangeTag	A



Storing Records Locally

Unarchiving system fields

Party	
Record ID	FDFE37608477
ChangeTag	A
PartyName	

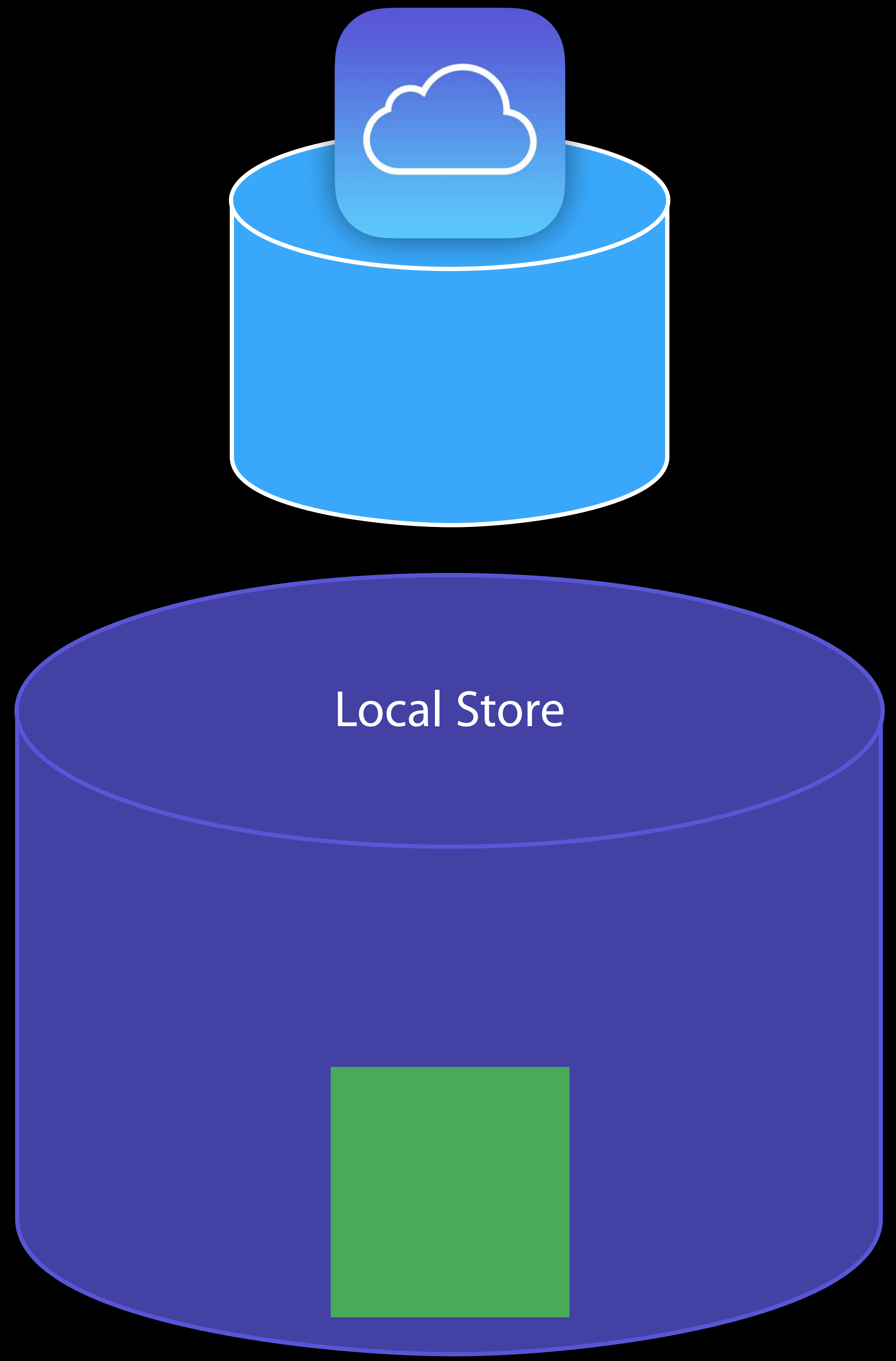


Local Store

Storing Records Locally

Unarchiving system fields

Party	
Record ID	FDFE37608477
ChangeTag	A
PartyName	WWDC Bash

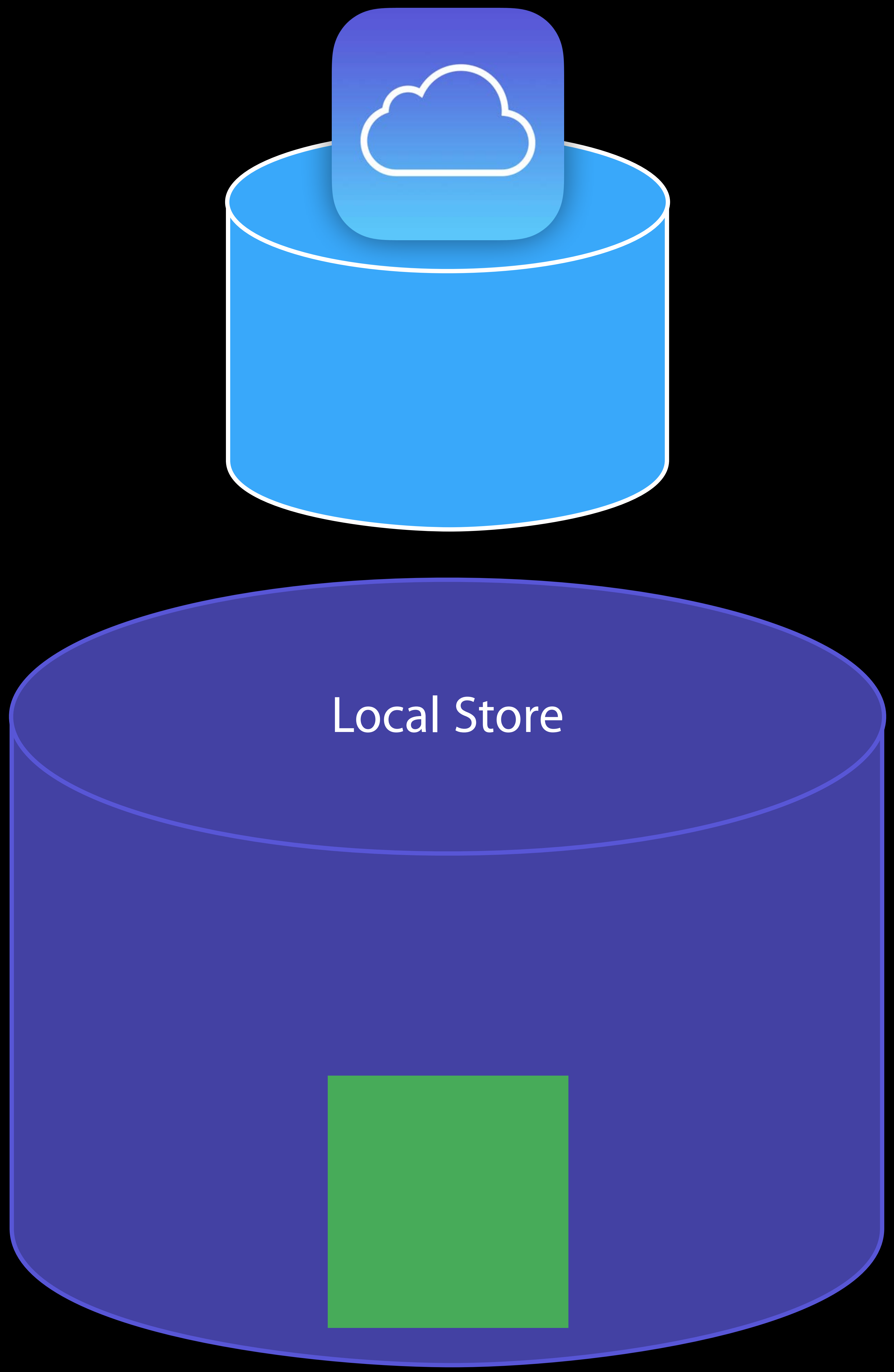


Local Store

Storing Records Locally

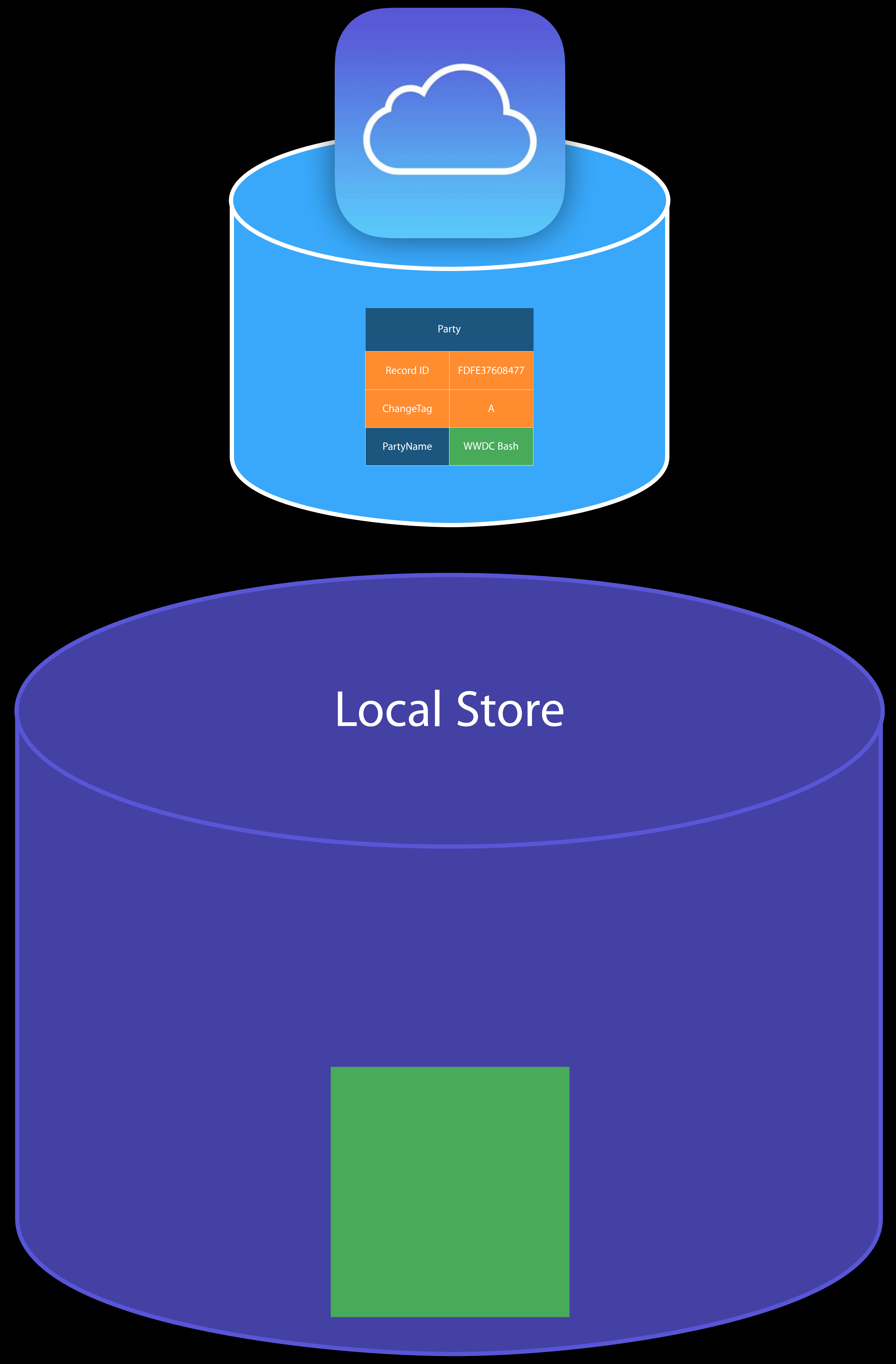
Unarchiving system fields

Party	
Record ID	FDFE37608477
ChangeTag	A
PartyName	WWDC Bash



Storing Records Locally

Unarchiving system fields



Party	
Record ID	FDFE37608477
ChangeTag	A
PartyName	WWDC Bash

Local Store

Efficiently Fetching Changes

Efficiently Fetching Changes

```
CKFetchRecordChangesOperation
```

Efficiently Fetching Changes

`CKFetchRecordChangesOperation`

Notifications using `CKSubscription`

Efficiently Fetching Changes

`CKFetchRecordChangesOperation`

Notifications using `CKSubscription`

- Silent notifications

Subscriptions

Subscription Recap

What is it?

Subscription Recap

What is it?

Persistent queries on the server

Subscription Recap

What is it?

Persistent queries on the server

Remote notification per relevant change

Subscription Recap

What is it?

Persistent queries on the server

Remote notification per relevant change

- Query subscriptions

Subscription Recap

What is it?

Persistent queries on the server

Remote notification per relevant change

- Query subscriptions
- Zone subscriptions

Subscription Recap

Setup

Subscription Recap

Setup

- APS capability managed from Developer Portal

Subscription Recap

Setup

- APS capability managed from Developer Portal
- `aps-environment` entitlement

Subscription Recap

Setup

- APS capability managed from Developer Portal
- `aps-environment` entitlement
- Registration via `UIApplication`

Subscription Recap

Setup

- APS capability managed from Developer Portal
- `aps-environment` entitlement
- Registration via `UIApplication`

  `registerForRemoteNotifications()`

Subscription Recap

Setup

- APS capability managed from Developer Portal
- `aps-environment` entitlement
- Registration via `UIApplication`

```
registerForRemoteNotifications()

registerUserNotificationSettings(notificationSettings:
UIUserNotificationSettings)
```

CloudKit Push Priorities

CloudKit Push Priorities

High priority push if `CKNotificationInfo` sets:

CloudKit Push Priorities

High priority push if `CKNotificationInfo` sets:

- `alertBody`

CloudKit Push Priorities

High priority push if `CKNotificationInfo` sets:

- `alertBody`
- `shouldBadge`

CloudKit Push Priorities

High priority push if `CKNotificationInfo` sets:

- `alertBody`
- `shouldBadge`
- `soundName`

CloudKit Push Priorities

High priority push if `CKNotificationInfo` sets:

- `alertBody`
- `shouldBadge`
- `soundName`

Medium priority or “silent” otherwise

Silent Push Notifications

How do I get one?

Silent Push Notifications

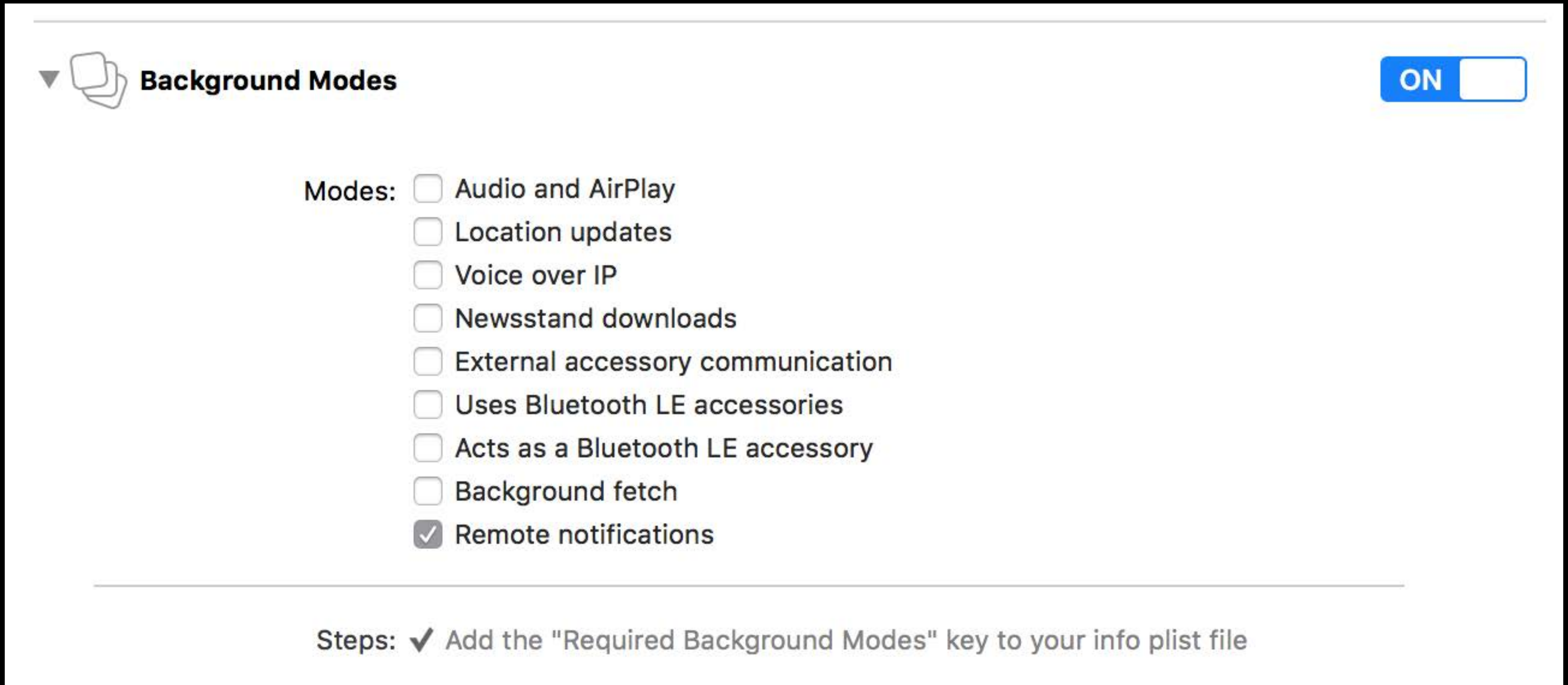
How do I get one?

1. Remote notification background mode

Silent Push Notifications

How do I get one?

1. Remote notification background mode



Background Modes ON

Modes:
□ Audio and AirPlay
□ Location updates
□ Voice over IP
□ Newsstand downloads
□ External accessory communication
□ Uses Bluetooth LE accessories
□ Acts as a Bluetooth LE accessory
□ Background fetch
☑ Remote notifications

Steps: ✓ Add the "Required Background Modes" key to your info plist file

Silent Push Notifications

How do I get one?

Silent Push Notifications

How do I get one?

2.

```
func application(application: UIApplication,
didReceiveRemoteNotification: [NSObject : AnyObject],
      fetchCompletionHandler: (UIBackgroundFetchResult) -> Void) {
...
}
```

Silent Push Notifications

How do I get one?

Silent Push Notifications

How do I get one?

3. `CKNotificationInfo`

Silent Push Notifications

How do I get one?

3. `CKNotificationInfo`

- `shouldSendContentAvailable = true`

Silent Push Notifications

How do I get one?

3. `CKNotificationInfo`

- `shouldSendContentAvailable = true`
- No `alertBody`, `shouldBadge` or `soundName`

Silent Push Notifications

Silent Push Notifications

Silent Push Notifications

System opportune time

Silent Push Notifications

System opportune time

Push delivery is best-effort

Silent Push Notifications

System opportune time

Push delivery is best-effort

- May get dropped or coalesced

Silent Push Notifications

Silent Push Notifications

Sync notification collection

Silent Push Notifications

Sync notification collection

- `CKFetchNotificationChangesOperation`

Silent Push Notifications

Sync notification collection

- `CKFetchNotificationChangesOperation`

Background task API:

Silent Push Notifications

Sync notification collection

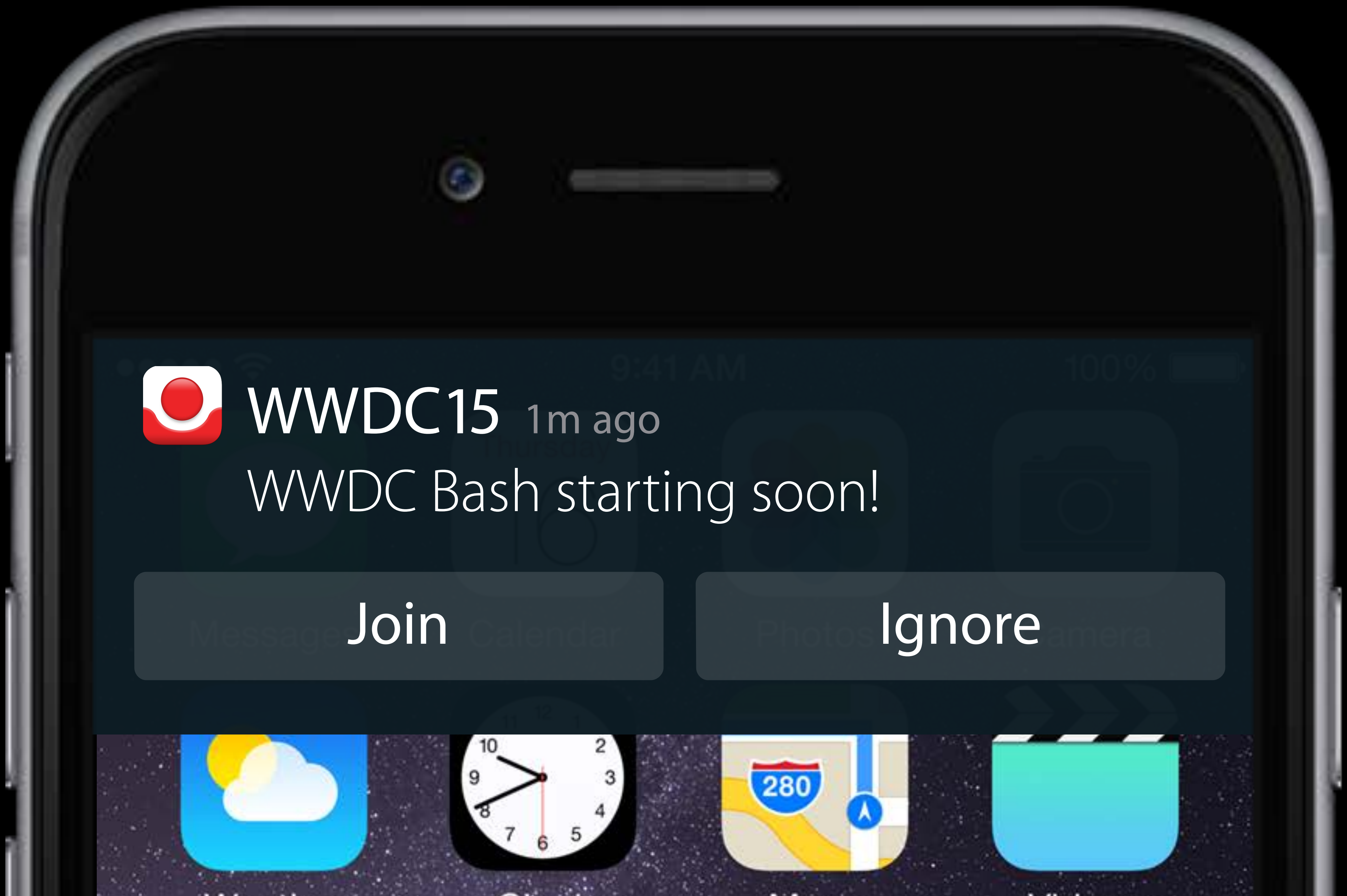
- `CKFetchNotificationChangesOperation`

Background task API:

```
class UIApplication : UIResponder {
   func beginBackgroundTaskWithName(taskName: String?,
           expirationHandler: (() -> Void)?) -> UIBackgroundTaskIdentifier
…
}
```

Interactive Notifications

Interactive Notifications



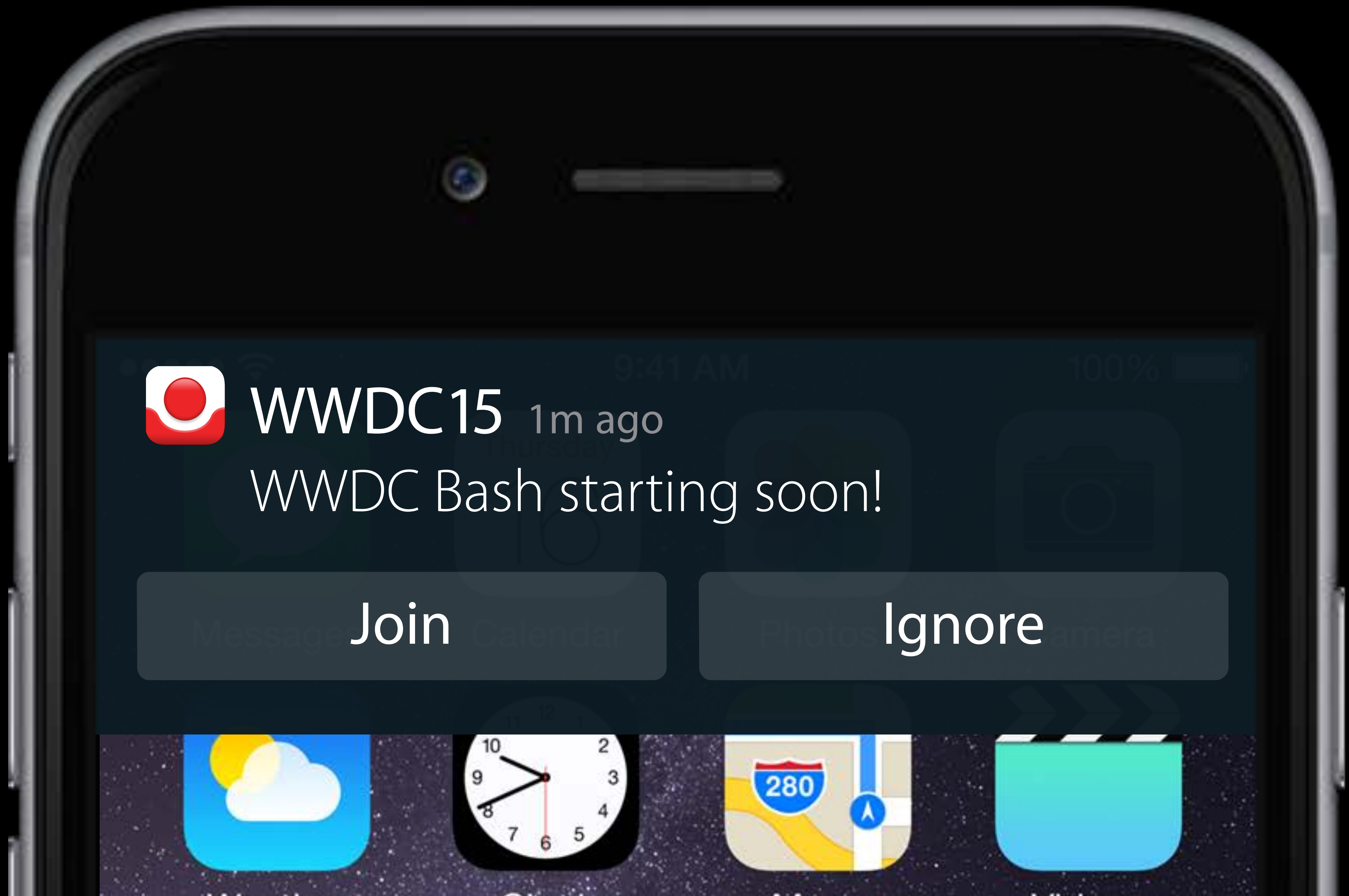
WWDC15 1m ago

WWDC Bash starting soon!

Interactive Notifications

NEW

```
class CKNotificationInfo : NSObject, …{
   var category: String?
…
}
```



WWDC15 1m ago

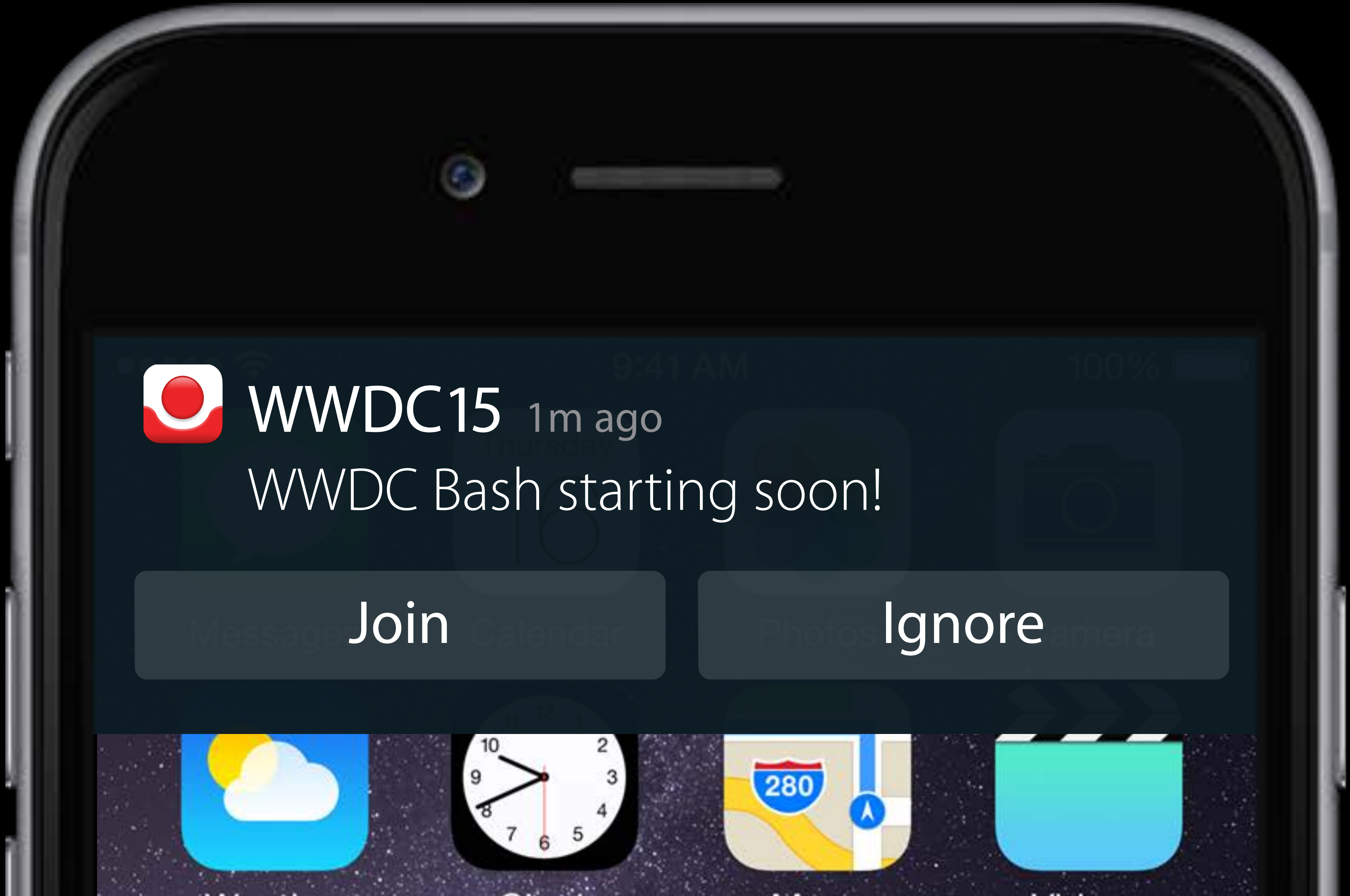
WWDC Bash starting soon!

Interactive Notifications

NEW

```
class CKNotificationInfo : NSObject, …{
   var category: String?
...
}
```

- See `UIMutableUserNotificationCategory`



WWDC15 1m ago

WWDC Bash starting soon!

CloudKit Performance Tips

Managing Dependent Tasks

Managing Dependent Tasks

CloudKit API is asynchronous

Managing Dependent Tasks

CloudKit API is asynchronous

Task dependencies are common

Managing Dependent Tasks

CloudKit API is asynchronous

Task dependencies are common

Goals:

Managing Dependent Tasks

CloudKit API is asynchronous

Task dependencies are common

Goals:

- Error handling

Managing Dependent Tasks

CloudKit API is asynchronous

Task dependencies are common

Goals:

- Error handling
- UI performance

Managing Dependent Tasks

CloudKit API is asynchronous

Task dependencies are common

Goals:

- Error handling
- UI performance
- Maintainability

Managing Dependent Tasks

Nesting convenience API calls

Managing Dependent Tasks

Nesting convenience API calls

```
database.fetchRecordWithID(recordID, { record, error in
    …
    database.fetchRecordWithID(otherRecordID, { otherRecord, otherError in
        …
        database.saveRecord(otherRecord, { savedRecord, anotherError in
            …
        })
    })
}
```

Managing Dependent Tasks

Nesting convenience API calls

```
database.fetchRecordWithID(recordID, { record, error in
    …
    database.fetchRecordWithID(otherRecordID, { otherRecord, otherError in
        …
        database.saveRecord(otherRecord, { savedRecord, anotherError in
            …
        })
    })
}
```

Managing Dependent Tasks

Making operations synchronous

Managing Dependent Tasks

Making operations synchronous

```
let sema = dispatch_semaphore_create(0)
database.fetchRecordWithID(recordID, { record, error in
    …
    dispatch_semaphore_signal(sema)
}
dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER)


database.fetchRecordWithID(someOtherRecordID, { otherRecord, otherError in
    …
})
```

Managing Dependent Tasks

Making operations synchronous

```
let sema = dispatch_semaphore_create(0)
database.fetchRecordWithID(recordID, { record, error in
    …
    dispatch_semaphore_signal(sema)
}
dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER)


database.fetchRecordWithID(someOtherRecordID, { otherRecord, otherError in
    …
})
```

Managing Dependent Tasks

NSOperation dependencies

Managing Dependent Tasks

NSOperation dependencies

```
class NSOperation : NSObject {

    func addDependency(op: NSOperation)
    func removeDependency(op: NSOperation)

    var dependencies: [NSOperation] { get }
…
}
```

Managing Dependent Tasks

NSOperation dependencies

Managing Dependent Tasks

NSOperation dependencies

```
let firstFetch = CKFetchRecordsOperation(…)
let secondFetch = CKFetchRecordsOperation(…)
…
secondFetch.addDependency(firstFetch)

let queue = NSOperationQueue()
queue.addOperations([firstFetch, secondFetch], waitUntilFinished: false)
```

NSOperation Quality of Service

NSOperation Quality of Service

```
class NSOperation : NSObject {
   @available(iOS 8.0, *)
    var qualityOfService: NSQualityOfService
...
}
```

NSOperation Quality of Service

```
class NSOperation : NSObject {
   @available(iOS 8.0, *)
    var qualityOfService: NSQualityOfService
…
}
```

```
enum NSQualityOfService : Int {
    case UserInteractive
    case UserInitiated
    case Utility
    case Background
    case Default
}
```

Discretionary Networking

Discretionary Networking

Non-user-initiated tasks

Discretionary Networking

Non-user-initiated tasks

- Example: pre-fetching content

Discretionary Networking

Non-user-initiated tasks

- Example: pre-fetching content

System waits for an opportune time:

Discretionary Networking

Non-user-initiated tasks

- Example: pre-fetching content

System waits for an opportune time:

- Wi-Fi vs. cellular

Discretionary Networking

Non-user-initiated tasks

- Example: pre-fetching content

System waits for an opportune time:

- Wi-Fi vs. cellular
- Power conditions

Discretionary Networking

Non-user-initiated tasks

- Example: pre-fetching content

System waits for an opportune time:

- Wi-Fi vs. cellular
- Power conditions

```
class CKOperation : NSOperation {
   var usesBackgroundSession: Bool
...
}
```

CKOperation Quality of Service

CKOperation Quality of Service

⊗ `usesBackgroundSession` is deprecated

CKOperation Quality of Service

☒ `usesBackgroundSession` is deprecated

✓ Use `qualityOfService`

CKOperation Quality of Service

☒ `usesBackgroundSession` is deprecated

✓ Use `qualityOfService`

Non-discretionary QoS:

CKOperation Quality of Service

✗ `usesBackgroundSession` is deprecated

✓ Use `qualityOfService`

Non-discretionary QoS:

`.UserInteractive` and `.UserInitiated`

CKOperation Quality of Service

✗ `usesBackgroundSession` is deprecated

✓ Use `qualityOfService`

Non-discretionary QoS:

`.UserInteractive` and `.UserInitiated`

Discretionary QoS:

CKOperation Quality of Service

✗ `usesBackgroundSession` is deprecated

✓ Use `qualityOfService`

Non-discretionary QoS:

`.UserInteractive` and `.UserInitiated`

Discretionary QoS:

`.Utility`

CKOperation Quality of Service

- ✗ `usesBackgroundSession` is deprecated
- ✓ Use `qualityOfService`

Non-discretionary QoS:

`.UserInteractive` and `.UserInitiated`

Discretionary QoS:

`.Utility`

`.Background` (default)

Advanced NSOperations

Where can I learn more?

Advanced NSOperations	Presidio	Friday 9:00AM

Summary

Summary

Error handling is vital

Summary

Error handling is vital

Batch requests

Summary

Error handling is vital

Batch requests

Schema trade-offs

Summary

Error handling is vital

Batch requests

Schema trade-offs

Configure CKOperations for performance

More Information

Documentation

CloudKit Resources

developer.apple.com/cloudkit

Technical Support

Apple Developer Forums

devforums.apple.com

Developer Technical Support

developer.apple.com/support/technical

General Inquiries

cloudkit@apple.com

Related Sessions

What’s New in CloudKit	Mission	Tuesday 3:30PM
CloudKit JS and Web Services	Pacific Heights	Wednesday 3:30PM

Related Lab

CloudKit Lab	Frameworks Lab D	Friday 9:00AM

WWDC15