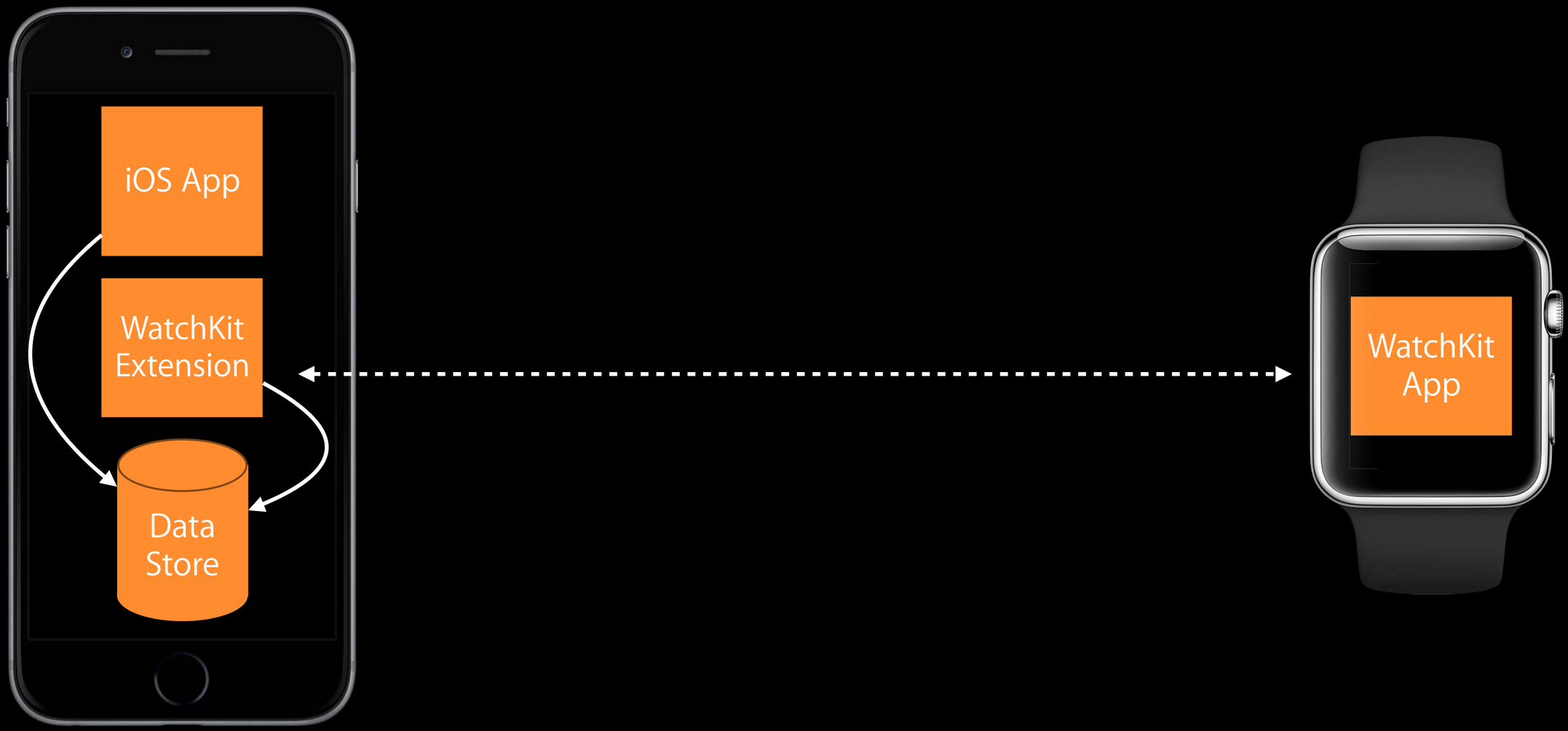
Introducing Watch Connectivity

Communication between devices

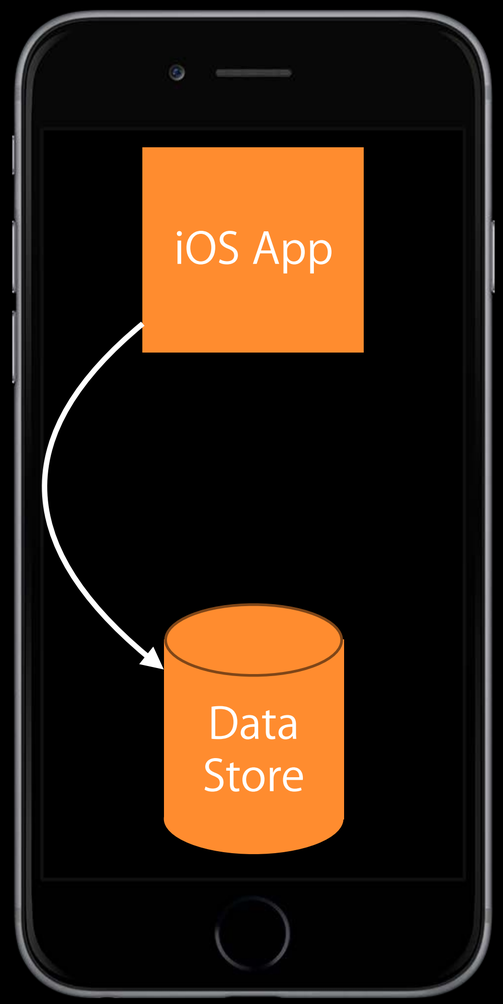
Session 713

Chris Jensen watchOS Engineer
Alex Ledwith watchOS Engineer

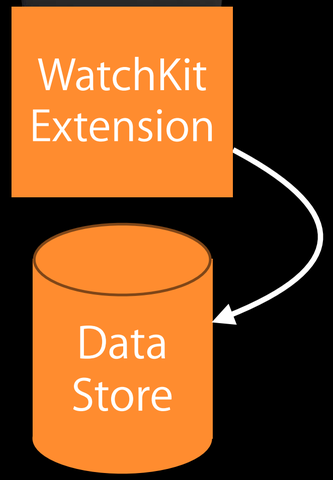
watchOS 1



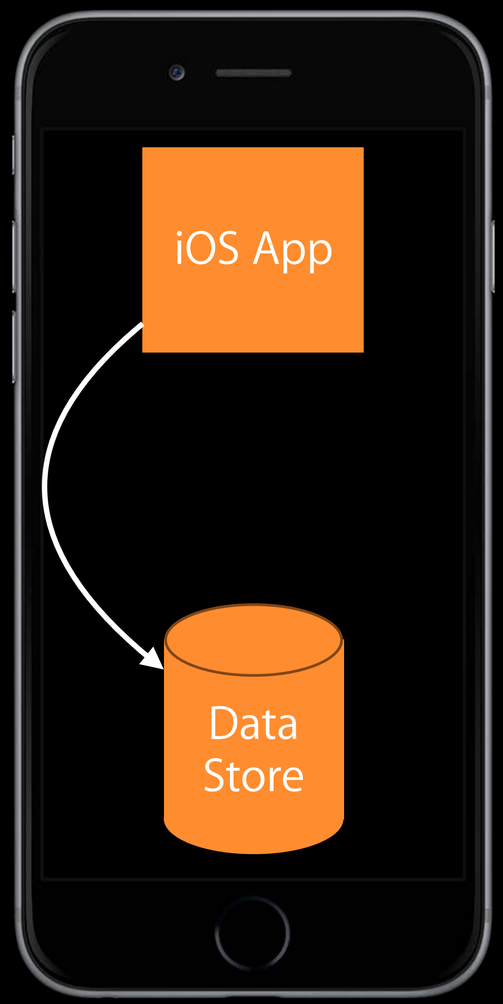
watchOS 2


iOS App
Data
Store


WatchKit
App


WatchKit
Extension
Data
Store

watchOS 2


iOS App
Data
Store


WatchKit
App
WatchKit
Extension
Data
Store

?

How do you get the data to the Apple Watch


NYC 1:09
8:21PM
11:00AM BRUNCH


10:09 AM
100%
Messages
Calendar
Monday
8
Photos
Camera
Weather
Clock
Maps
Videos
Wallet
Notes
Reminders
Stocks
iTunes Store
App Store
iBooks
News
Health
Settings
Phone
Safari
Mail
Music

NYC 1:09
8:21PM
11:00AM BRUNCH

WatchConnectivity

WatchConnectivity

NSURLSession

Example Apps with watchOS 2 Support

Social Networking App

Image Editing App

Weather App

Sports App

iOS Game

Run Tracker App

Surfing App

News App

WatchConnectivity

NSURLSession

Complications

WatchConnectivity

Setup

Setup

Always set your apps up to receive incoming WatchConnectivity content

Setup

Always set your apps up to receive incoming WatchConnectivity content

```
if (WCSession.isSupported()) {
```

Setup

Always set your apps up to receive incoming WatchConnectivity content

```
if (WCSession.isSupported()) {
    let session = WCSession.defaultSession()
```

Setup

Always set your apps up to receive incoming WatchConnectivity content

```
if (WCSession.isSupported()) {
    let session = WCSession.defaultSession()
    session.delegate = self // conforms to WCSessionDelegate
```

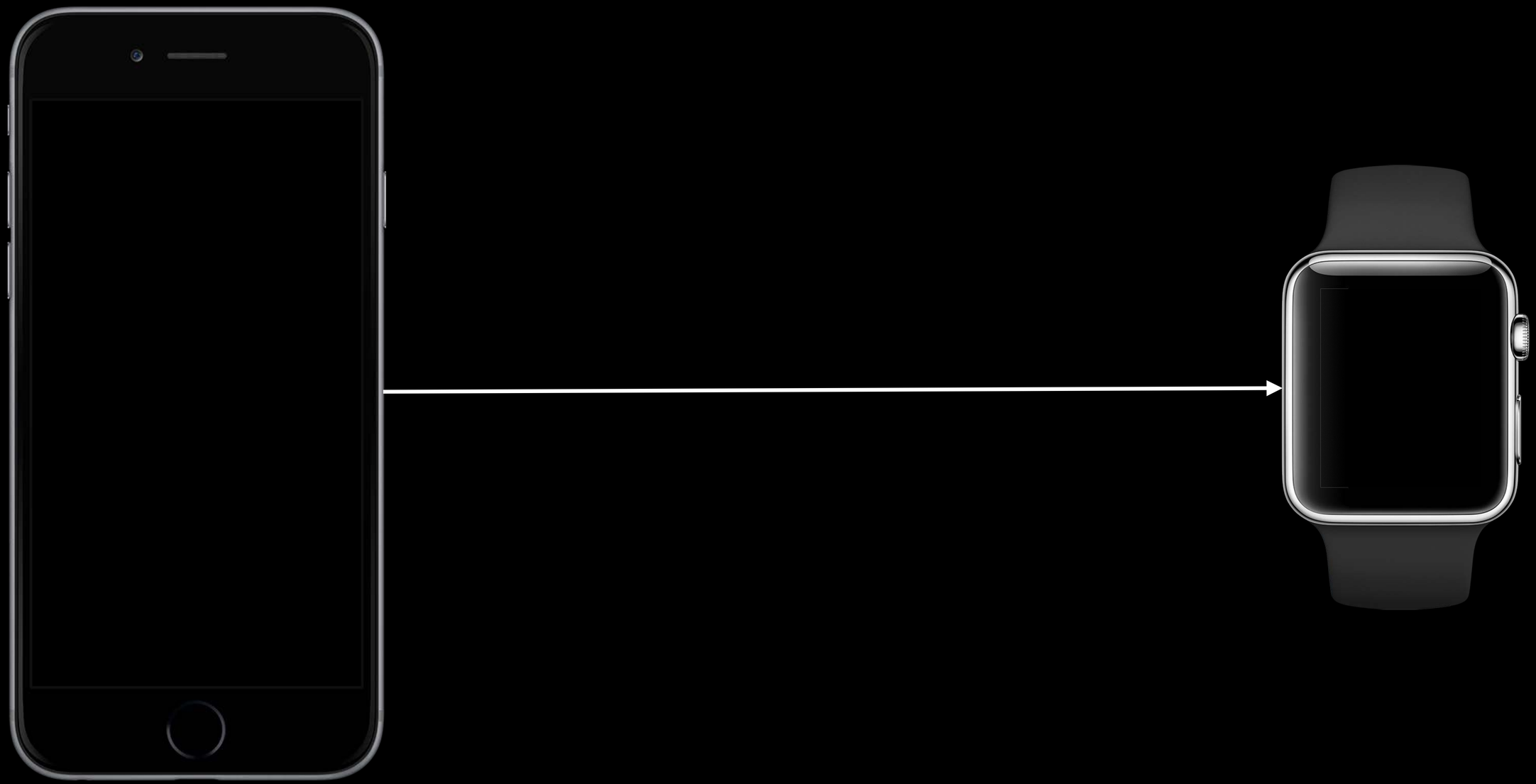
Setup

Always set your apps up to receive incoming WatchConnectivity content

```
if (WCSession.isSupported()) {
    let session = WCSession.defaultSession()
    session.delegate = self // conforms to WCSessionDelegate
    session.activateSession()
}
```

Session State

Session State



Session State

Session State


```
let session = WCSession.defaultSession()
session.delegate = self
session.activateSession()
```

Session State

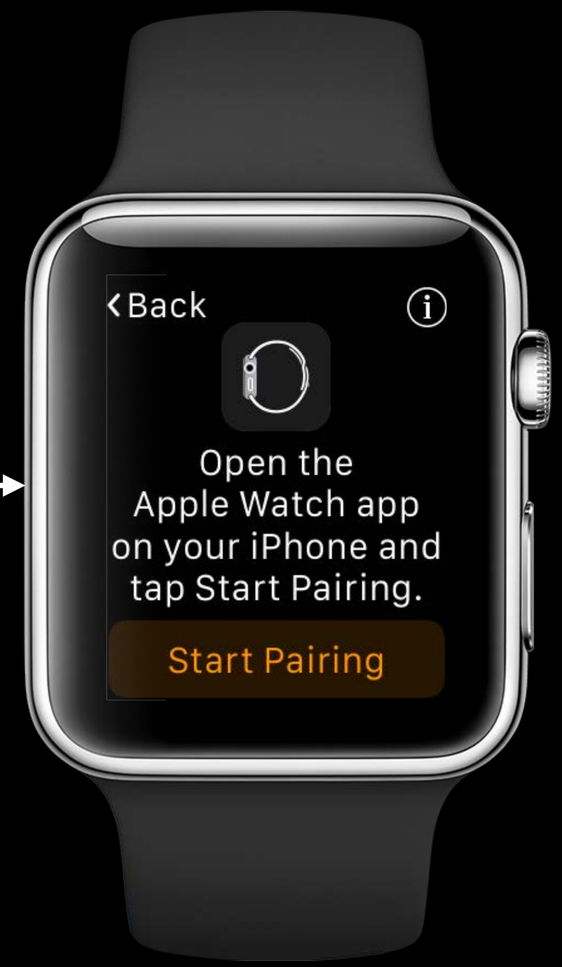

```
session.paired == false
```

Session State

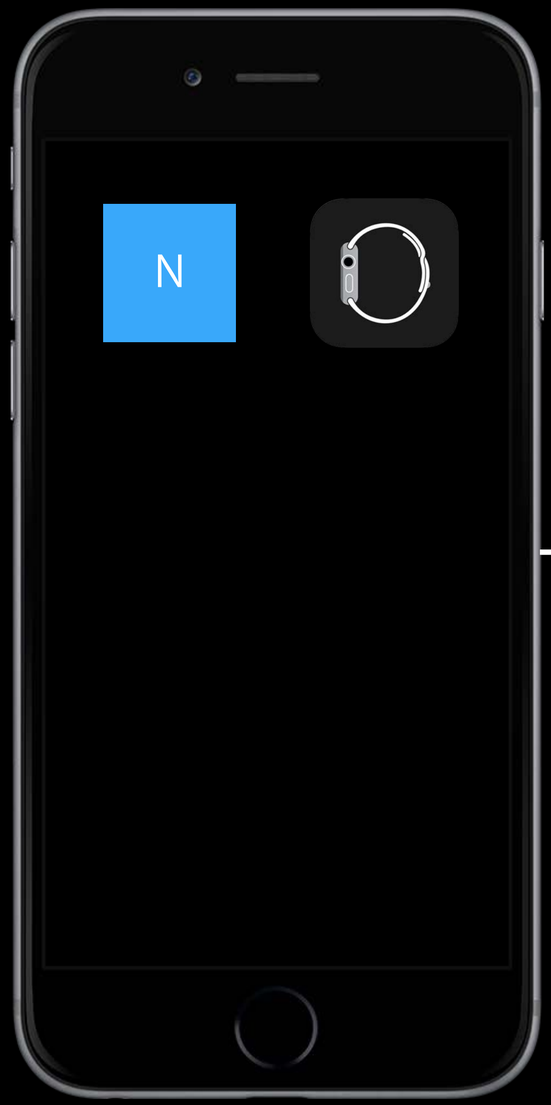

```
session.paired == false
```

Session State

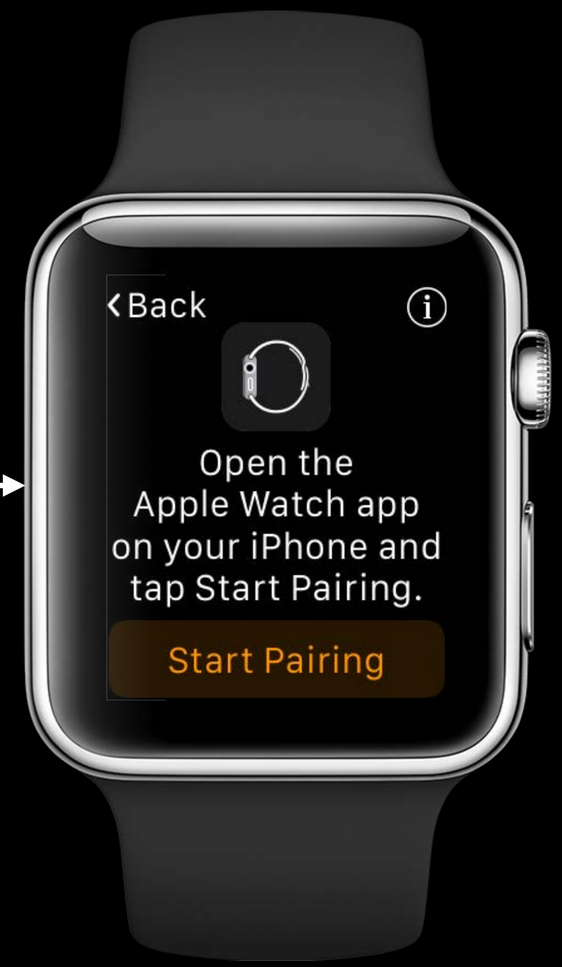
```
session.paired == false
```



Session State

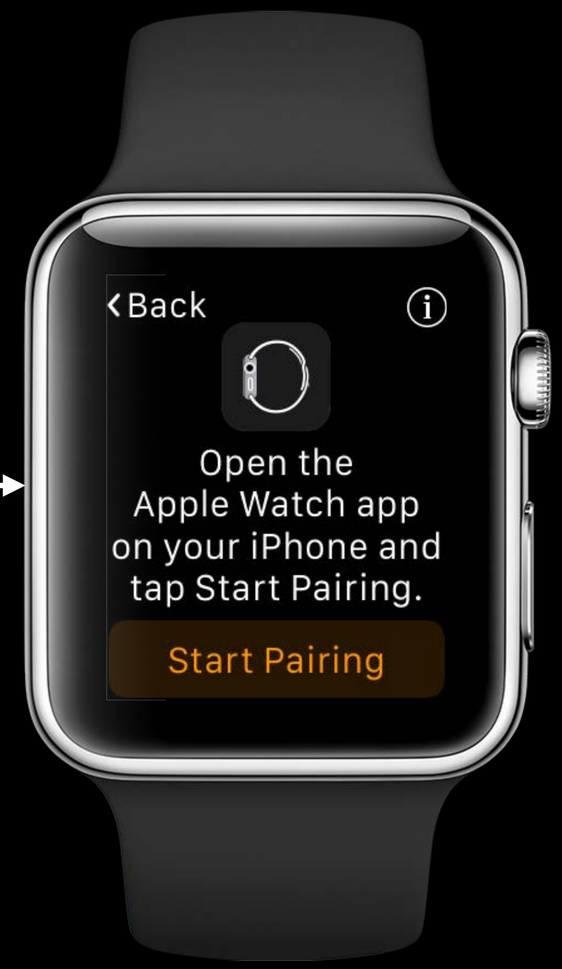



```
session.paired == false
```

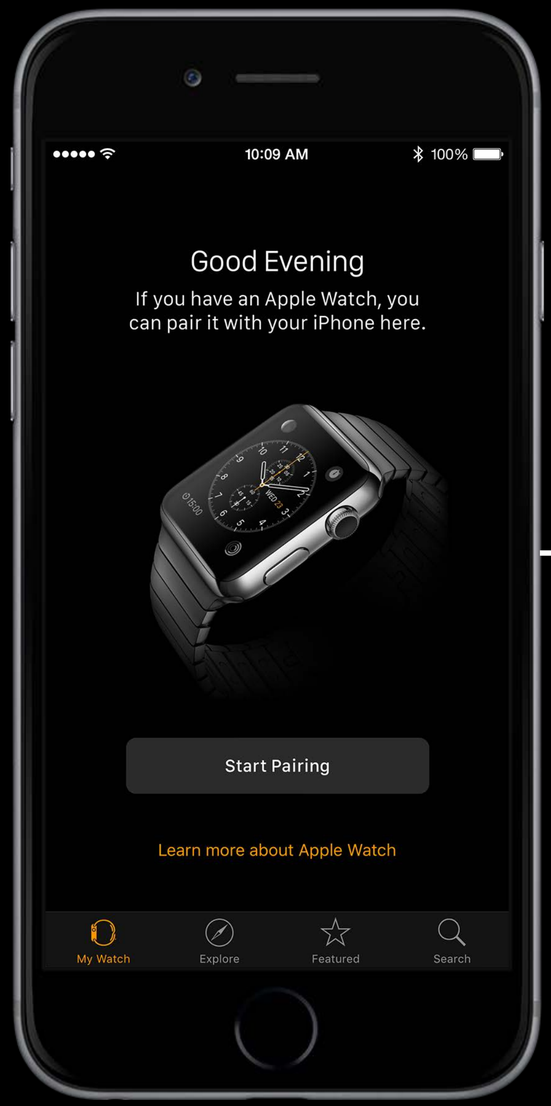


Session State

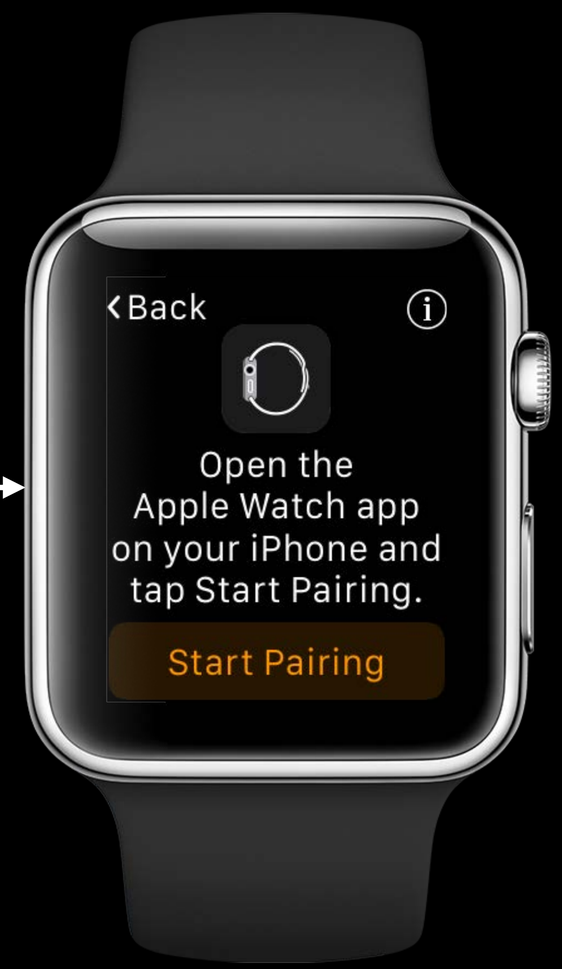
```
session.paired == false
```



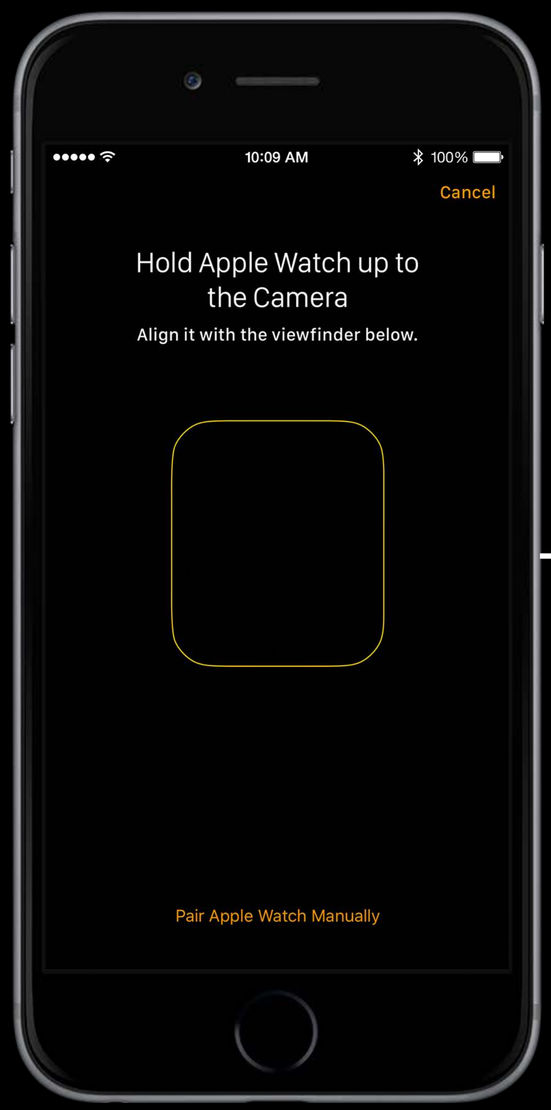
Session State



`session.paired == false`



Session State




```
session.paired == false
```

Session State

```
session.paired == false
```

Session State


```
session.paired == false
```

Session State


```
session.paired == false
```

```
func sessionWatchStateDidChange(_)
```

Session State


```
session.paired == true
```

```
func sessionWatchStateDidChange(_)
```

Session State


```
session.watchAppInstalled == false
```

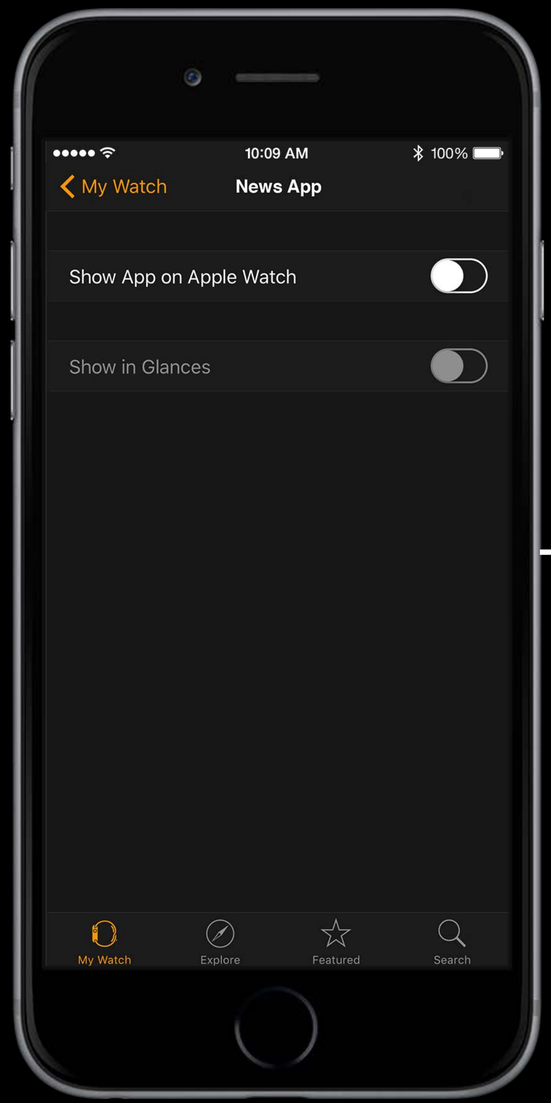
Session State

```
session.watchAppInstalled == false
```

Session State

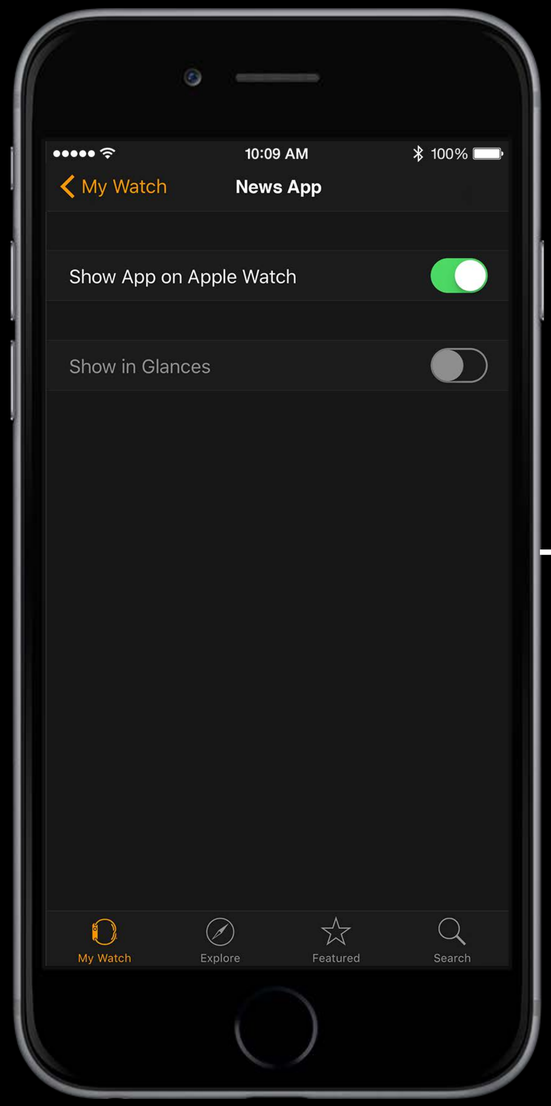
```
session.watchAppInstalled == false
```

Session State



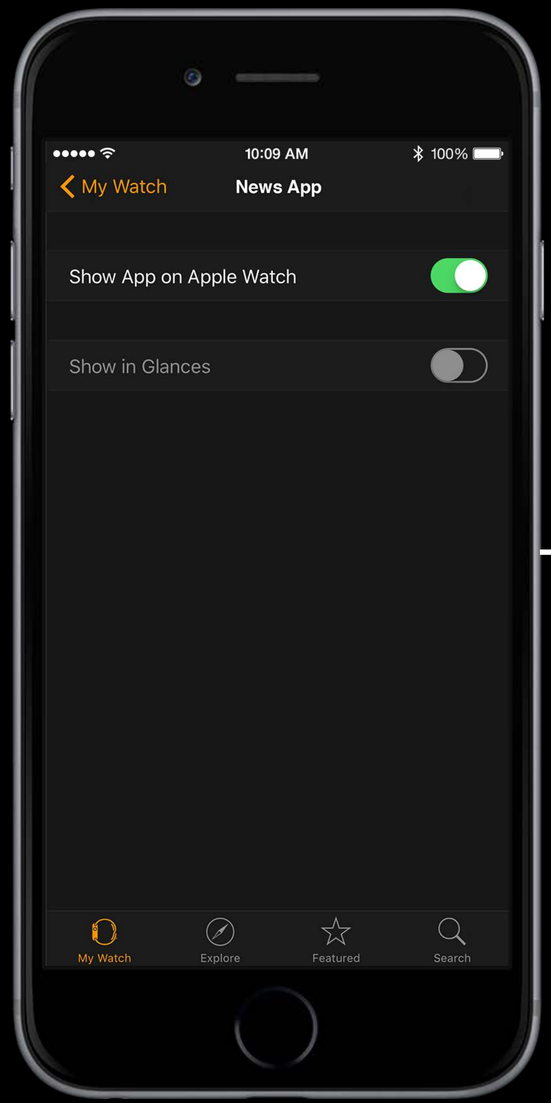

```
session.watchAppInstalled == false
```

Session State




```
session.watchAppInstalled == false
```

Session State




```
session.watchAppInstalled == false
```

Session State

```
session.watchAppInstalled == false
```

Session State


```
session.watchAppInstalled == false
```

Session State


```
session.watchAppInstalled == true
```

Session State


```
session.watchAppInstalled == true
```

```
session.watchDirectoryURL != nil
```

Session State

Watch directory

Session State

Watch directory

Directory and its contents' lifetime is tied to the `watchAppInstalled` property

Session State

Watch directory

Directory and its contents' lifetime is tied to the `watchAppInstalled` property

Use only for data relevant to the specific instance of your Watch app

Session State

Watch directory

Directory and its contents' lifetime is tied to the `watchAppInstalled` property

Use only for data relevant to the specific instance of your Watch app

- 'Last queued item' marker

Session State

Watch directory

Directory and its contents' lifetime is tied to the `watchAppInstalled` property

Use only for data relevant to the specific instance of your Watch app

- 'Last queued item' marker
- Preferences

Session State

Watch directory

Directory and its contents' lifetime is tied to the `watchAppInstalled` property

Use only for data relevant to the specific instance of your Watch app

- 'Last queued item' marker
- Preferences
- Files queued for transfer

Session State


```
session.complicationEnabled == false
```

Session State


```
session.complicationEnabled == false
```

Session State


```
session.complicationEnabled == true
```

Communication

Alex Ledwith

Communication

Categories

Communication

Categories

Background transfers

Communication

Categories

Background transfers

- Content not needed immediately

Communication

Categories

Background transfers

- Content not needed immediately
- OS intelligently transfers content

Communication

Categories

Background transfers

- Content not needed immediately
- OS intelligently transfers content

Interactive messaging

Communication

Categories

Background transfers

- Content not needed immediately
- OS intelligently transfers content

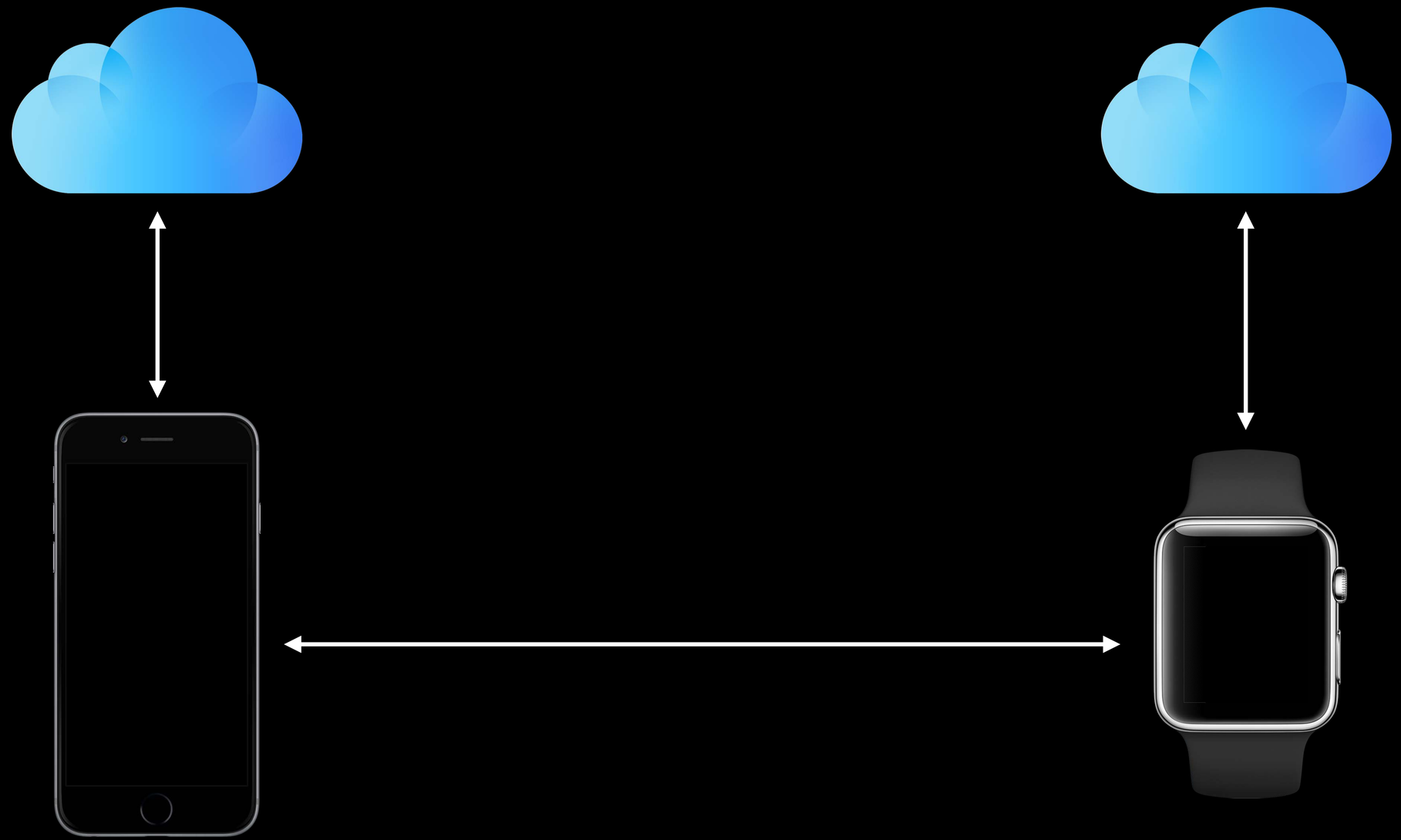
Interactive messaging

- Live communication

Background Transfers

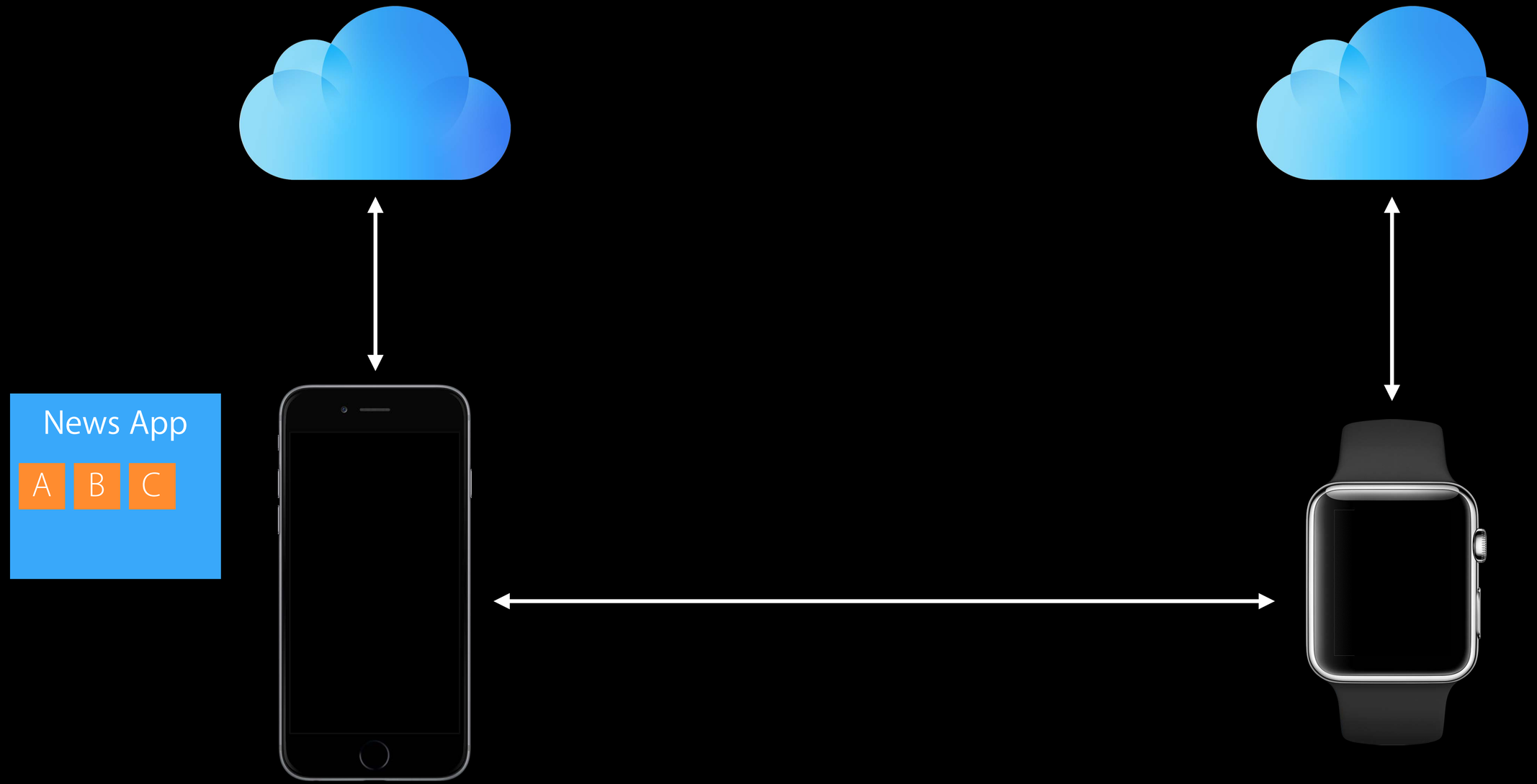
Background Transfers

Content and user interaction



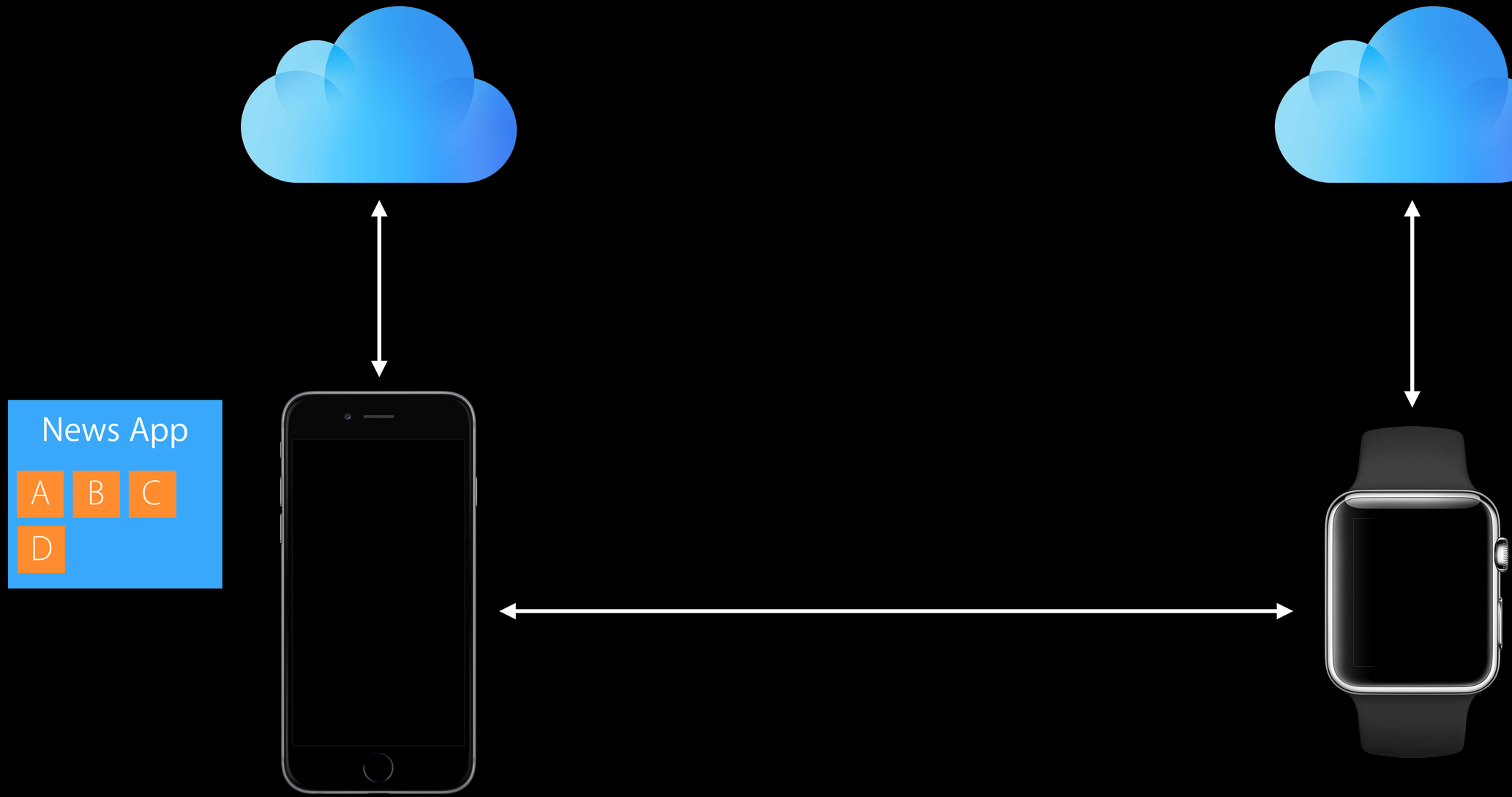
Background Transfers

Content and user interaction



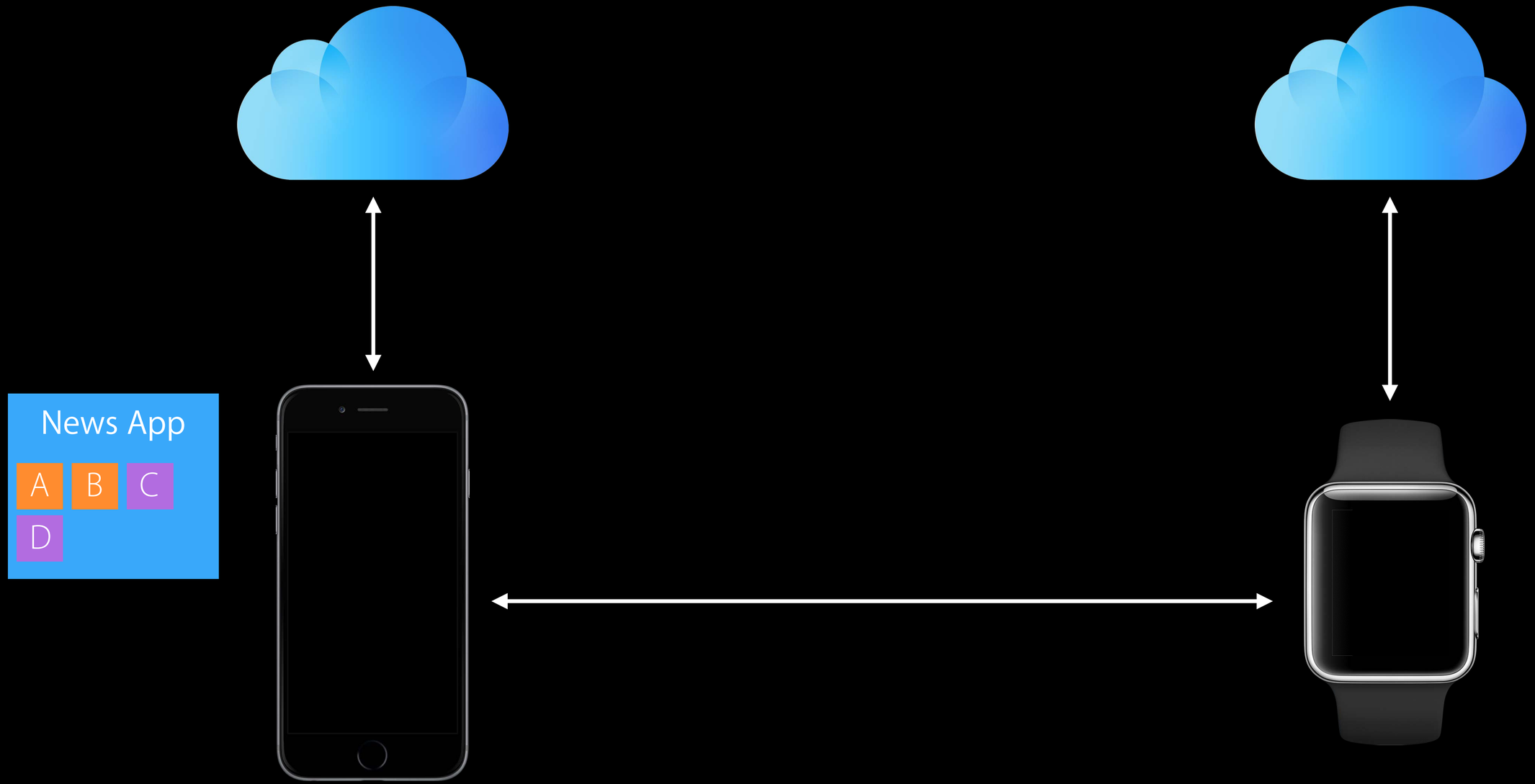
Background Transfers

Content and user interaction



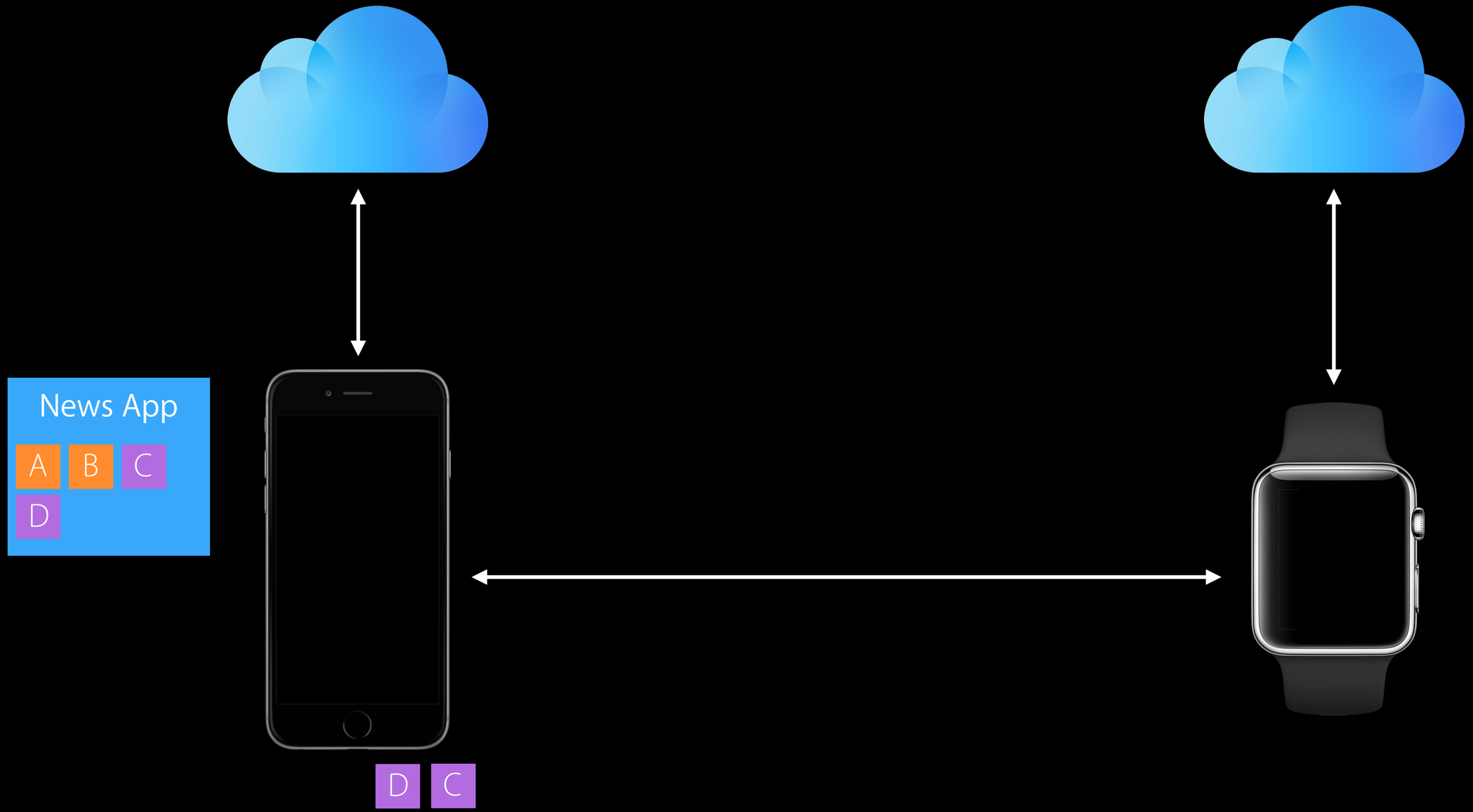
Background Transfers

Content and user interaction



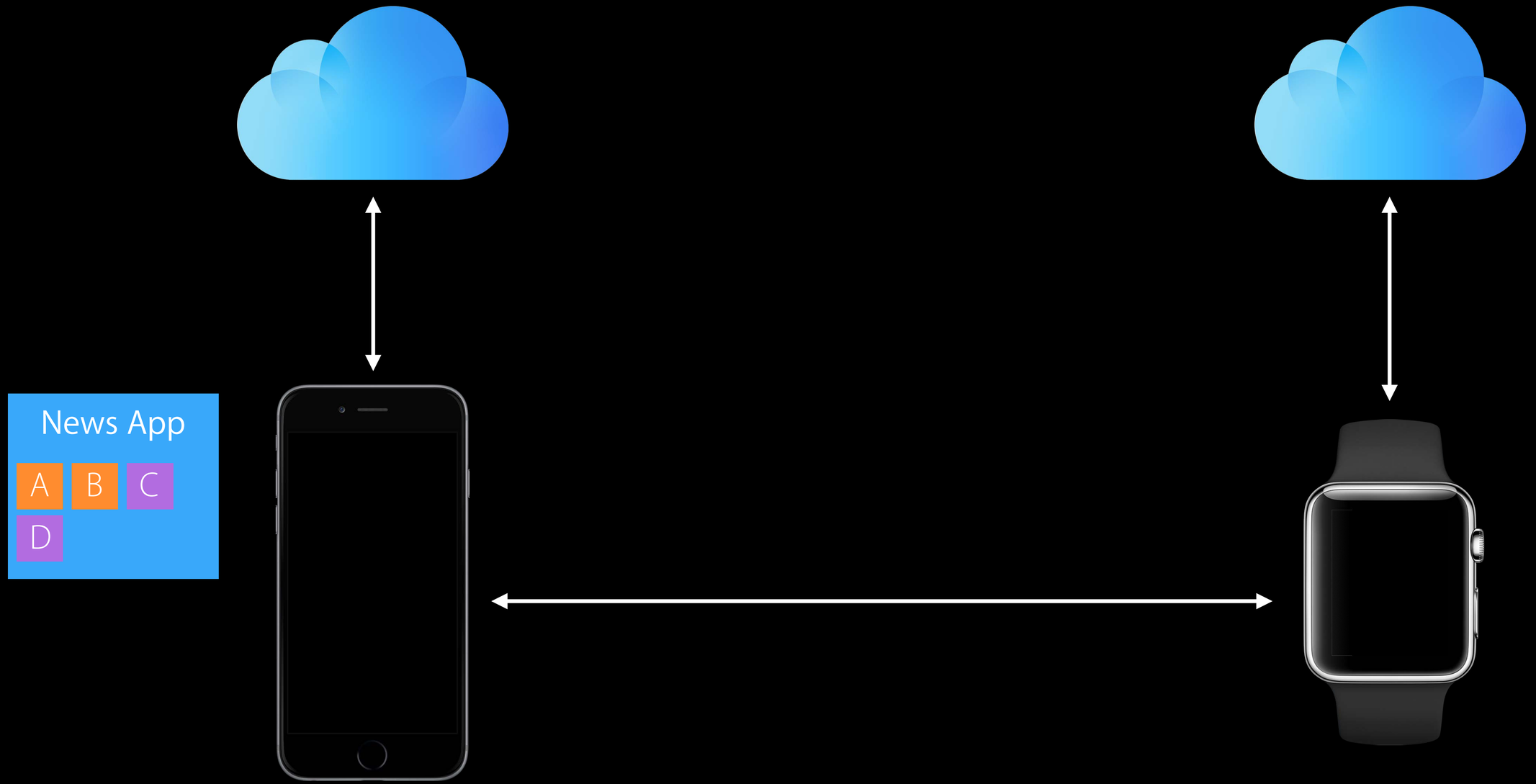
Background Transfers

Content and user interaction



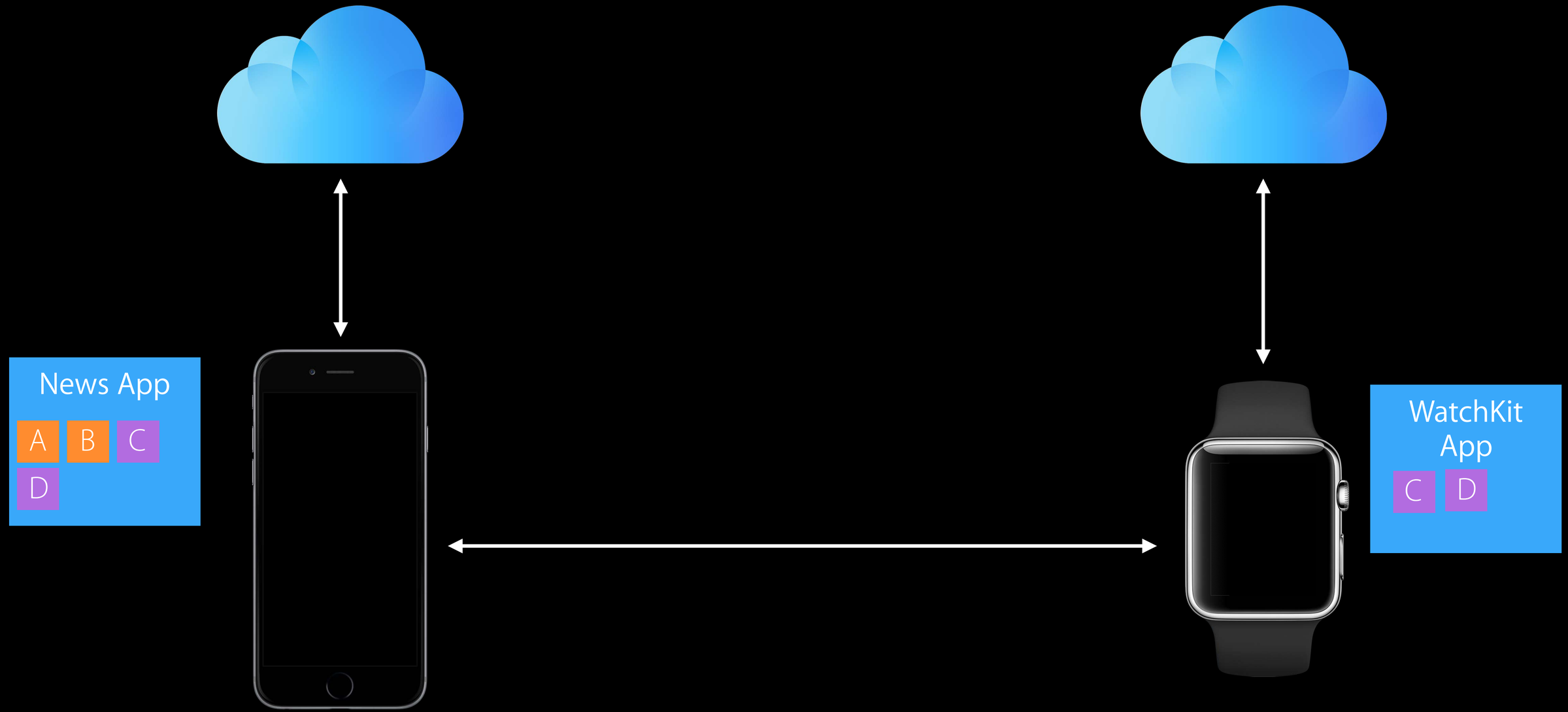
Background Transfers

Content and user interaction



Background Transfers

Content and user interaction



Background Transfers

Introduction

Background Transfers

Introduction

Queue up content

Background Transfers

Introduction

Queue up content

OS transfers content

Background Transfers

Introduction

Queue up content

OS transfers content

- Sending side can exit

Background Transfers

Introduction

Queue up content

OS transfers content

- Sending side can exit
- Pick opportune time

Background Transfers

Introduction

Queue up content

OS transfers content

- Sending side can exit
- Pick opportune time
- Delivers on receiver next launch

Background Transfers

Introduction

Queue up content

OS transfers content

- Sending side can exit
- Pick opportune time
- Delivers on receiver next launch

Recommended

Background Transfers

Introduction

Queue up content

OS transfers content

- Sending side can exit
- Pick opportune time
- Delivers on receiver next launch

Recommended

- Most information not needed immediately

Background Transfers

Types

Background Transfers

Types

Application context

Background Transfers

Types

Application context

User info transfer

Background Transfers

Types

Application context

User info transfer

File transfer

Background Transfers

Application context

Background Transfers

Application context

`applicationContext`

`receivedApplicationContext`

Background Transfers

Application context

`updateApplicationContext(_:)`

`applicationContext`

`receivedApplicationContext`

Background Transfers

Application context

`updateApplicationContext(_:)`

`applicationContext`

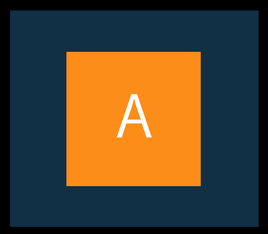
`receivedApplicationContext`

Background Transfers

Application context

`updateApplicationContext(_:)`

`applicationContext`



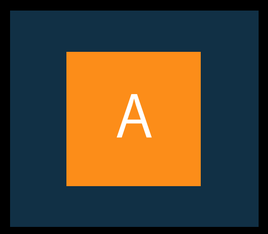
`receivedApplicationContext`

Background Transfers

Application context

`updateApplicationContext(_:)`

`applicationContext`



`receivedApplicationContext`

Background Transfers

Application context

`updateApplicationContext(_:)`

`applicationContext`

`receivedApplicationContext`

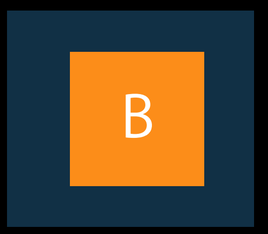
Background Transfers

Application context

`updateApplicationContext(_:)`

`applicationContext`

`receivedApplicationContext`



Background Transfers

Application context

`updateApplicationContext(_:)`

`applicationContext`

`receivedApplicationContext`

Background Transfers

Application context: Code

Background Transfers

Application context: Code

```
do {
    let context = // Create context dictionary with latest state
```

Background Transfers

Application context: Code

```
do {
    let context = // Create context dictionary with latest state
    try WCSession.defaultSession().updateApplicationContext(context)
```

Background Transfers

Application context: Code

```
do {
    let context = // Create context dictionary with latest state
    try WCSession.defaultSession().updateApplicationContext(context)
} catch {
    // Handle any errors
}
```

Background Transfers

Application context: Code

```
// Receiver Callback
func session(session: WCSession, didReceiveApplicationContext:
applicationContext: [String : AnyObject]) {
    // Handle application context dictionary
}
```

Background Transfers

Application context

Background Transfers

Application context

Most interesting/relevant content

Background Transfers

Application context

Most interesting/relevant content

Overriding behavior

Background Transfers

Application context

Most interesting/relevant content

Overriding behavior

Dictionary

Background Transfers

Application context

Most interesting/relevant content

Overriding behavior

Dictionary

- Property list types

Background Transfers

Application context

Most interesting/relevant content

Overriding behavior

Dictionary

- Property list types

Recommended use cases

Background Transfers

Application context

Most interesting/relevant content

Overriding behavior

Dictionary

- Property list types

Recommended use cases

- Many Apple Watch apps

Background Transfers

Application context

Most interesting/relevant content

Overriding behavior

Dictionary

- Property list types

Recommended use cases

- Many Apple Watch apps
- Glances

Background Transfers

User info transfer

Background Transfers

User info transfer

Outstanding User Info Transfers

Background Transfers

User info transfer

Outstanding User Info Transfers

L1 L2

Background Transfers

User info transfer


```
transferUserInfo(_:)
```

Outstanding User Info Transfers

L1 L2

Background Transfers

User info transfer

`transferUserInfo(_:)`

L3

Outstanding User Info Transfers

L1

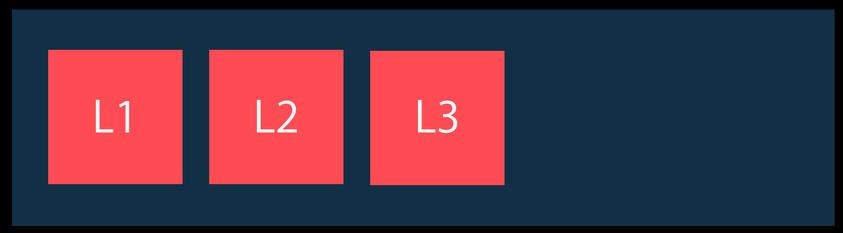
L2

Background Transfers

User info transfer

`transferUserInfo(_:)`

Outstanding User Info Transfers



Background Transfers

User info transfer

L1 L2 L3


```
transferUserInfo(_:)
```

Outstanding User Info Transfers

Background Transfers

User info transfer

`transferUserInfo(_:)`

Outstanding User Info Transfers

Background Transfers

User info transfer: Code

Background Transfers

User info transfer: Code

```
let userInfo = // Create dictionary of userInfo
```

Background Transfers

User info transfer: Code

```
let userInfo = // Create dictionary of userInfo
let userInfoTransfer = WCSession.defaultSession().transferUserInfo(userInfo)
```

Background Transfers

User info transfer: Code

```
let userInfo = // Create dictionary of userInfo
let userInfoTransfer = WCSession.defaultSession().transferUserInfo(userInfo)
```

```
let transfers = WCSession.defaultSession().outstandingUserInfoTransfers()
```

Background Transfers

User info transfer: Code

```
// Receiver Callback
func session(session: WCSession, didReceiveUserInfo userInfo: [String :
AnyObject]) {
    // Handle incoming user info dictionary
}
```

Background Transfers

User info transfer

Background Transfers

User info transfer

Queue user infos (dictionaries)

Background Transfers

User info transfer

Queue user infos (dictionaries)

- Property list types

Background Transfers

User info transfer

Queue user infos (dictionaries)

- Property list types

In memory content

Background Transfers

User info transfer

Queue user infos (dictionaries)

- Property list types

In memory content

Access to outstanding content in queue

Background Transfers

File transfer

Background Transfers

File transfer

Outstanding File Transfers

~/Documents/Inbox/

Background Transfers

File transfer

Outstanding File Transfers

P2 P1

~/Documents/Inbox/

Background Transfers

File transfer

```
transferFile(_: metadata:)
```

Outstanding File Transfers

P2 P1

~/Documents/Inbox/

Background Transfers

File transfer

```
transferFile(_: metadata:)
```

P3

Outstanding File Transfers

P2 P1

~/Documents/Inbox/

Background Transfers

File transfer

```
transferFile(_: metadata:)
```

Outstanding File Transfers

P3 P2 P1

~/Documents/Inbox/

Background Transfers

File transfer

```
transferFile(_: metadata:)
```

Outstanding File Transfers

~/Documents/Inbox/

P3 P2 P1

Background Transfers

File transfer

```
transferFile(_: metadata:)
```

Outstanding File Transfers

~/Documents/Inbox/

Background Transfers

File transfer: Code

Background Transfers

File transfer: Code

```
let url = // Retrieve URL of file
```

Background Transfers

File transfer: Code

```
let url = // Retrieve URL of file
let metadata = // Create dictionary of metadata
```

Background Transfers

File transfer: Code

```
let url = // Retrieve URL of file
let metadata = // Create dictionary of metadata
let fileTransfer = WCSession.defaultSession().transferFile(url,
metadata:metadata)
```

Background Transfers

File transfer: Code

```
let url = // Retrieve URL of file
let metadata = // Create dictionary of metadata
let fileTransfer = WCSession.defaultSession().transferFile(url,
metadata:metadata)
```

```
let transfers = WCSession.defaultSession().outstandingFileTransfers()
```

Background Transfers

File transfer: Delegate callback

Background Transfers

File transfer: Delegate callback

```
// Receiver Callback
func session(session: WCSession, didReceiveFile file: WCSessionFile) {
    // Handle file URL and metadata in WCSessionFile object
}
```

Background Transfers

File transfer: Delegate callback

WCSessionFile

```
// Receiver Callback
func session(session: WCSession, didReceiveFile file: WCSessionFile) {
    // Handle file URL and metadata in WCSessionFile object
}
```

Background Transfers

File transfer: Delegate callback

WCSessionFile

- File URL

```
// Receiver Callback
func session(session: WCSession, didReceiveFile file: WCSessionFile) {
    // Handle file URL and metadata in WCSessionFile object
}
```

Background Transfers

File transfer: Delegate callback

WCSessionFile

- File URL
- Metadata

```
// Receiver Callback
func session(session: WCSession, didReceiveFile file: WCSessionFile) {
    // Handle file URL and metadata in WCSessionFile object
}
```

Background Transfers

File transfer: Delegate callback

WCSessionFile

- File URL
- Metadata

Move file to take control

```
// Receiver Callback
func session(session: WCSession, didReceiveFile file: WCSessionFile) {
    // Handle file URL and metadata in WCSessionFile object
}
```

Background Transfers

File transfer: Delegate callback

WCSessionFile

- File URL
- Metadata

Move file to take control

- File deleted from inbox

```
// Receiver Callback
func session(session: WCSession, didReceiveFile file: WCSessionFile) {
    // Handle file URL and metadata in WCSessionFile object
}
```

Background Transfers

File transfer

Background Transfers

File transfer

Very similar to user info transfer

Background Transfers

File transfer

Very similar to user info transfer

Queue files

Background Transfers

File transfer

Very similar to user info transfer

Queue files

Access to outstanding content in queue

Background Transfers

File transfer

Very similar to user info transfer

Queue files

Access to outstanding content in queue

Additional metadata

Background Transfers

File transfer

Very similar to user info transfer

Queue files

Access to outstanding content in queue

Additional metadata

- Small

Background Transfers

File transfer

Very similar to user info transfer

Queue files

Access to outstanding content in queue

Additional metadata

- Small
- Property list types

Interactive Messaging

Interactive Messaging

Introduction

Interactive Messaging

Reachability

Interactive Messaging

Reachability

Other app available

Interactive Messaging

Reachability

Other app available

Required for messaging

Interactive Messaging

Reachability

Other app available

Required for messaging

Property on WCSession

```
WCSession.defaultSession().reachable
```

Interactive Messaging

Reachability: iPhone

Interactive Messaging

Reachability: iPhone

Devices connected

Interactive Messaging

Reachability: iPhone

Devices connected

Watch app foreground

Interactive Messaging

Reachability: iPhone

Devices connected

Watch app foreground


```
session.reachable == true
```

Interactive Messaging

Reachability: Apple Watch

Interactive Messaging

Reachability: Apple Watch

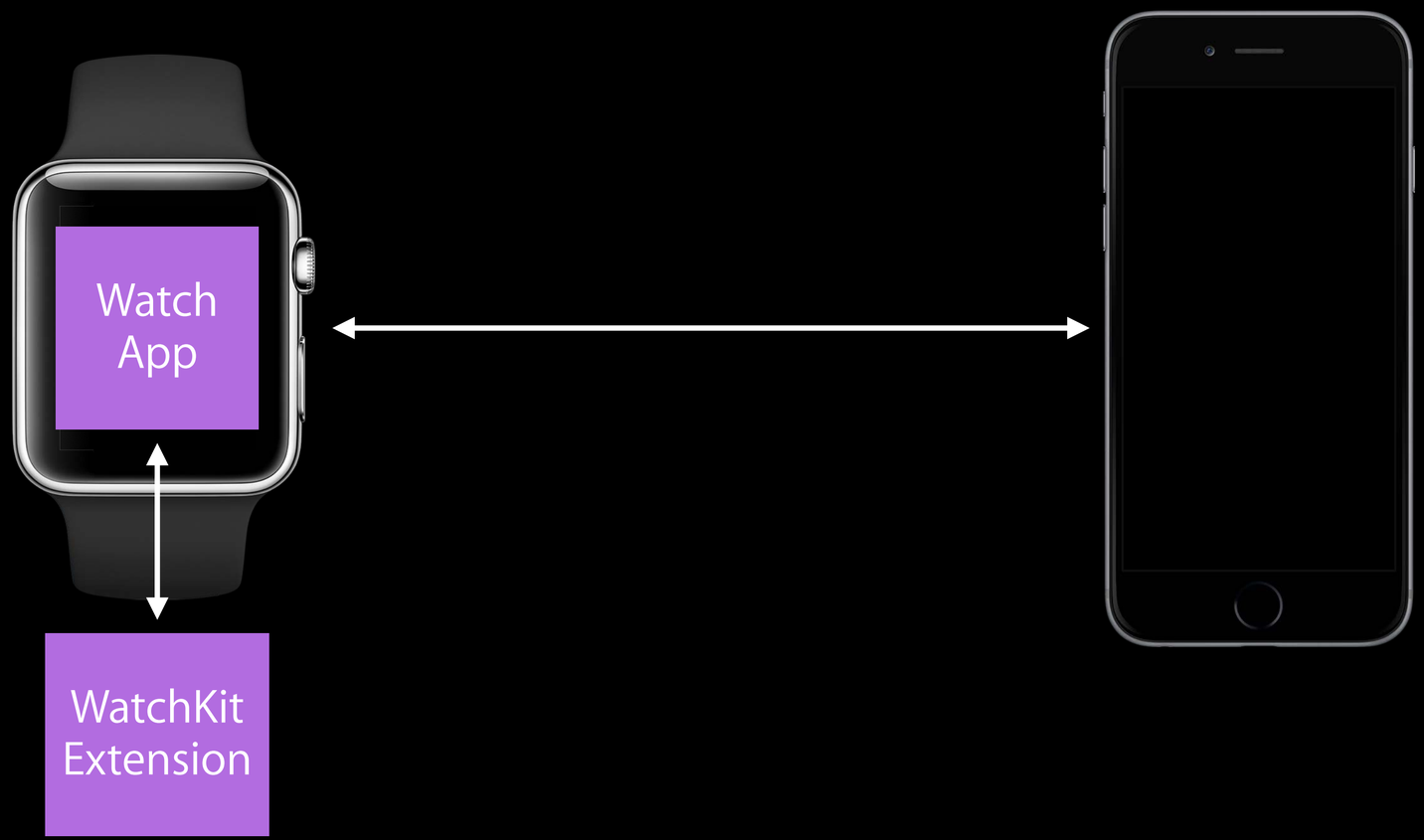
Devices connected

Interactive Messaging

Reachability: Apple Watch

Devices connected

WatchKit extension foreground

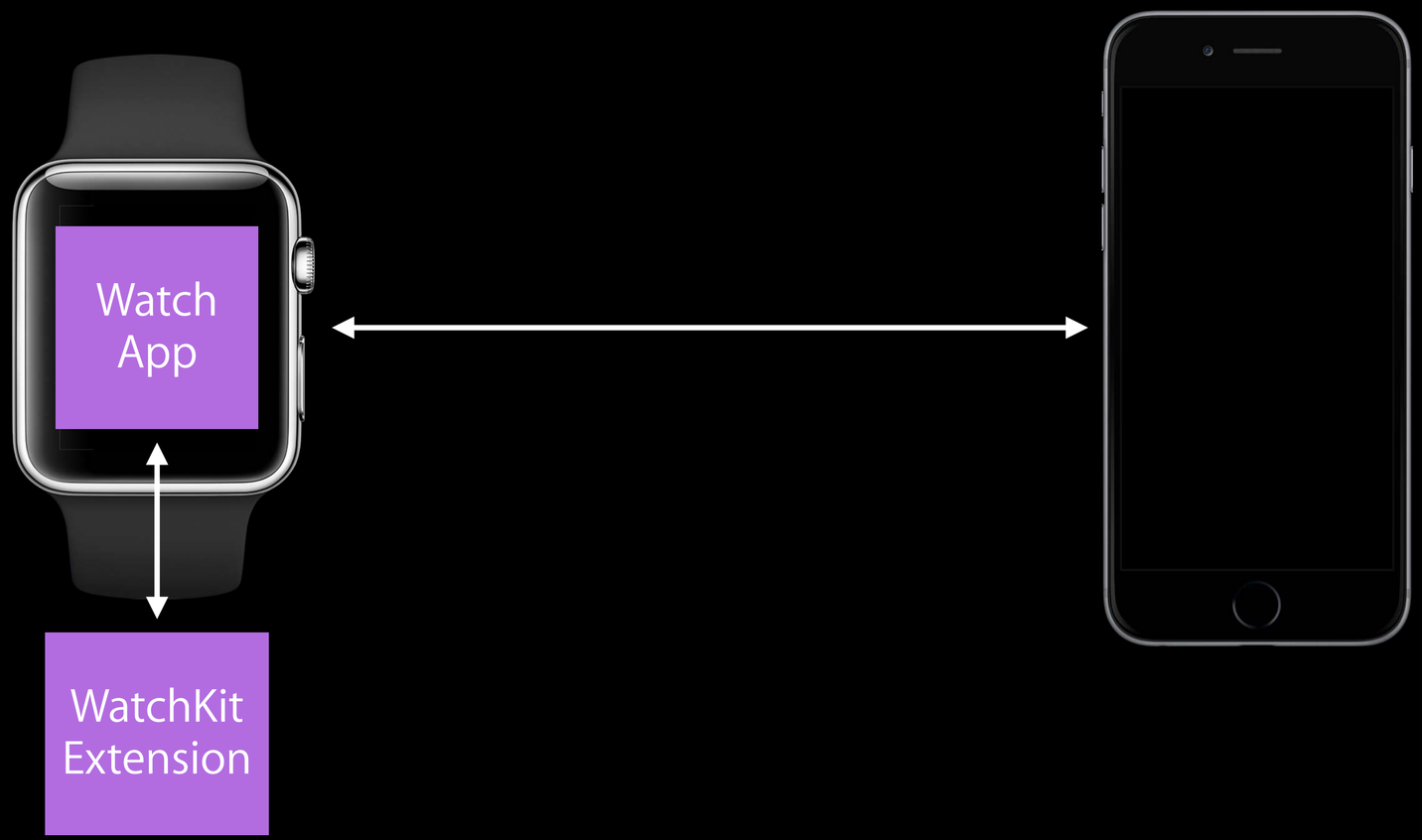


Interactive Messaging

Reachability: Apple Watch

Devices connected

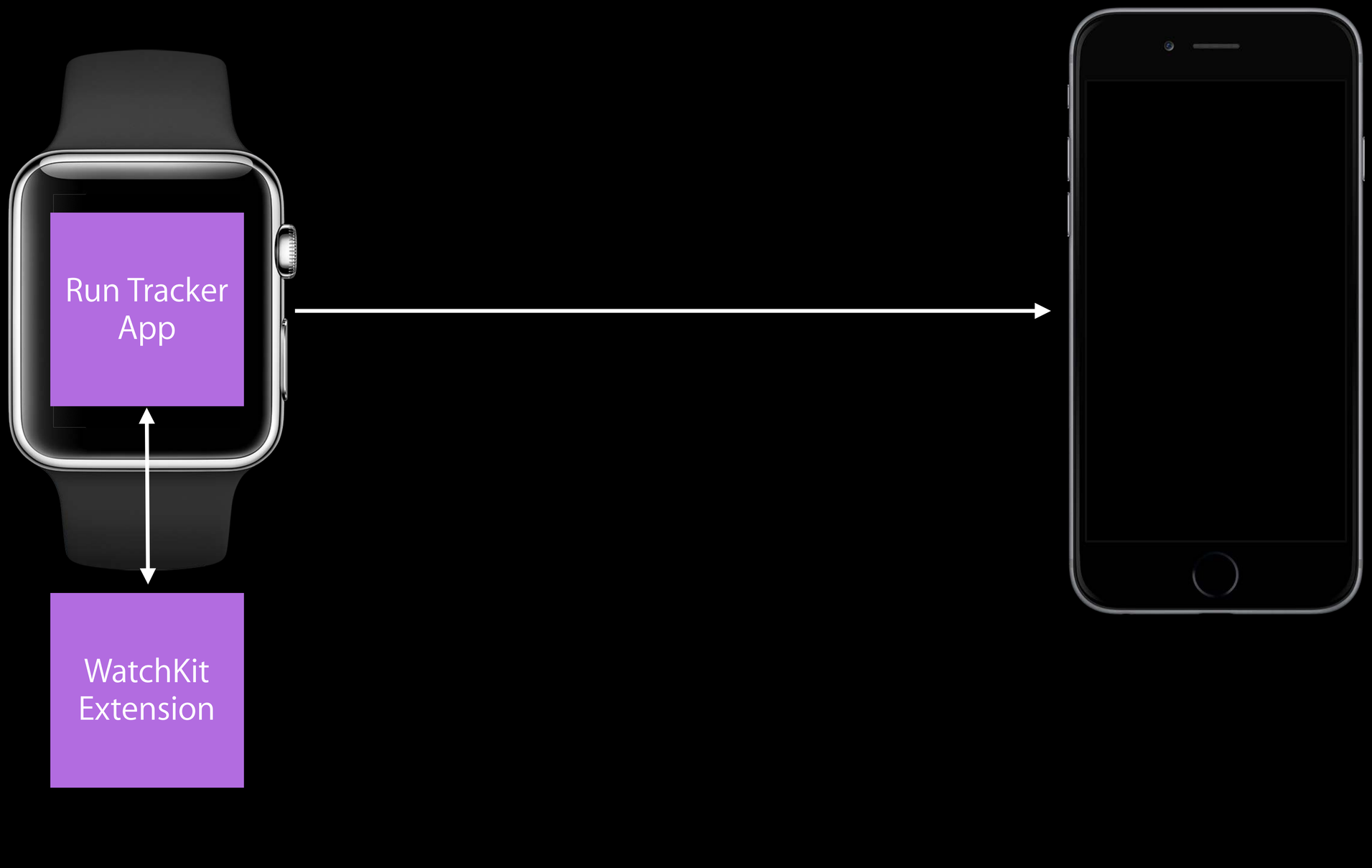
WatchKit extension foreground




```
session.reachable == true
```

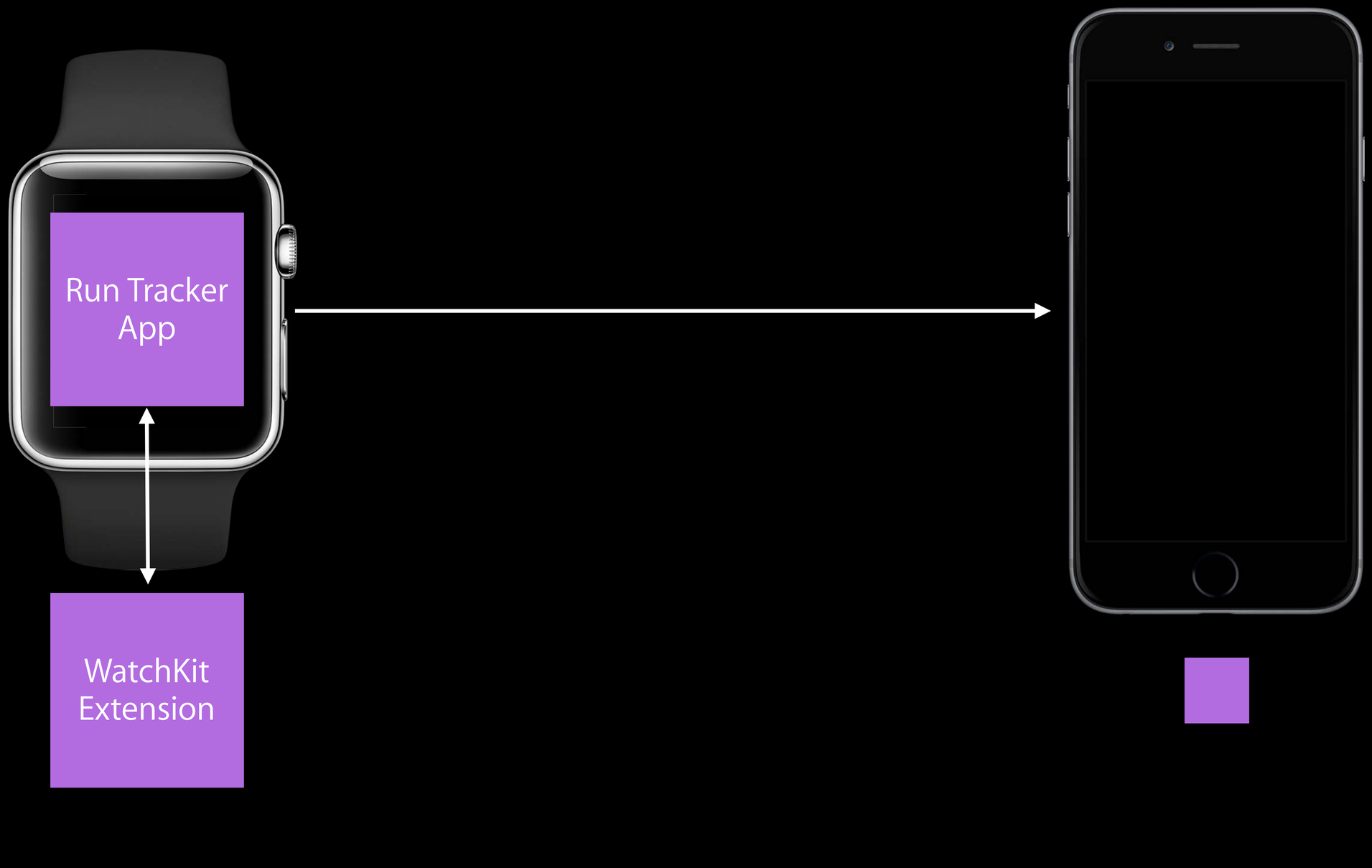
Interactive Messaging

Background launching: iOS app only



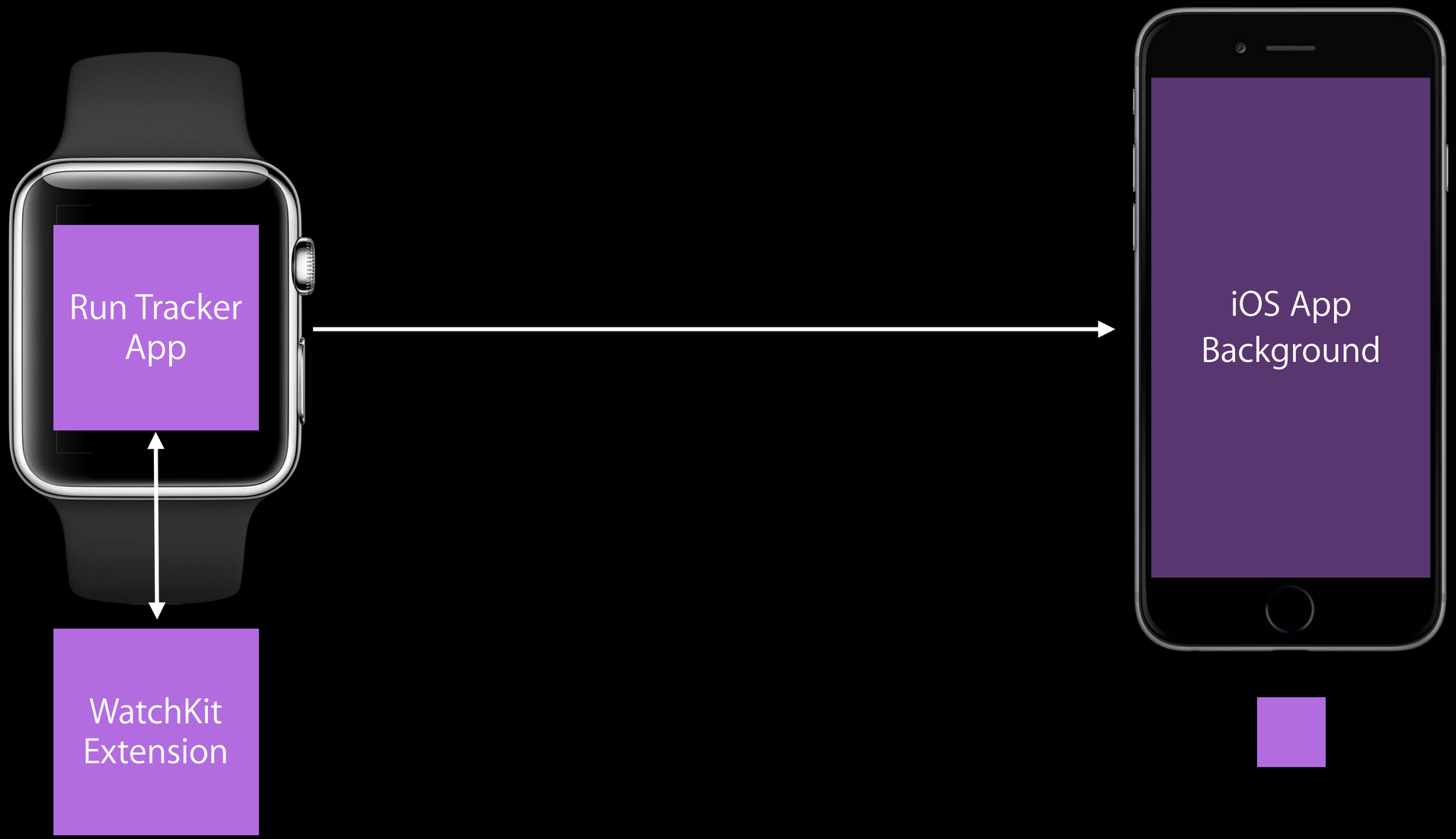
Interactive Messaging

Background launching: iOS app only



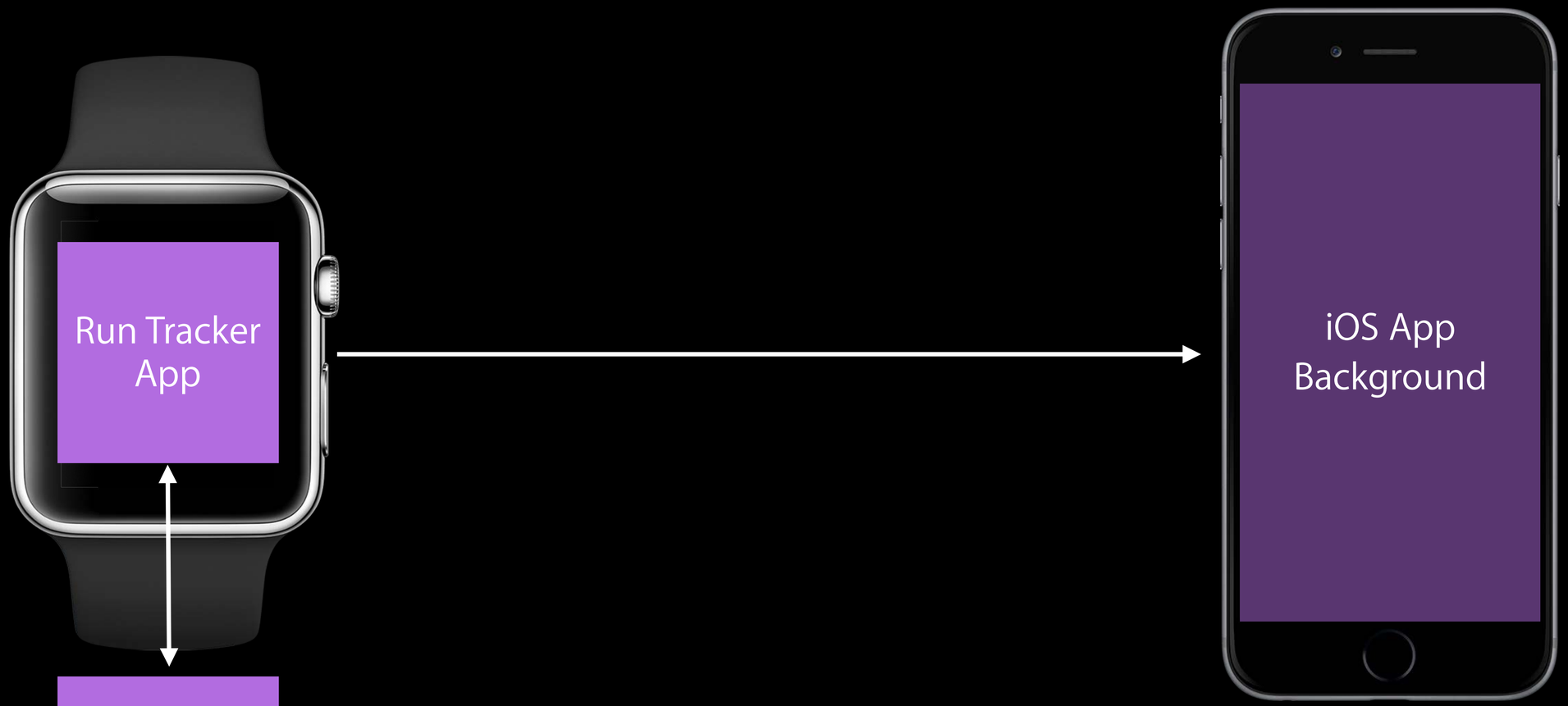
Interactive Messaging

Background launching: iOS app only



Interactive Messaging

Background launching: iOS app only



Interactive Messaging

Types

Interactive Messaging

Types

Dictionary

Interactive Messaging

Types

Dictionary

```
func sendMessage(message:, replyHandler:, errorHandler:)
```

Interactive Messaging

Types

Dictionary

- Property list types

```
func sendMessage(message:, replyHandler:, errorHandler:)
```

Interactive Messaging

Types

Dictionary

- Property list types

Data

```
func sendMessage(message:, replyHandler:, errorHandler:)
```

Interactive Messaging

Types

Dictionary

- Property list types

Data

```
func sendMessage(message:, replyHandler:, errorHandler:)

func sendMessageData(data:, replyHandler:, errorHandler:)
```

Interactive Messaging

Types

Dictionary

- Property list types

Data

- Custom data

```
func sendMessage(message:, replyHandler:, errorHandler:)

func sendMessageData(data:, replyHandler:, errorHandler:)
```

Interactive Messaging

Types

Dictionary

- Property list types

Data

- Custom data
- Own serialization

```
func sendMessage(message:, replyHandler:, errorHandler:)

func sendMessageData(data:, replyHandler:, errorHandler:)
```

Interactive Messaging

Replying

Interactive Messaging

Replying

```
func sendMessage(message:, replyHandler:, errorHandler:)

func sendMessageData(data:, replyHandler:, errorHandler:)
```

Interactive Messaging

Replying

Optional handler

```
func sendMessage(message:, replyHandler:, errorHandler:)

func sendMessageData(data:, replyHandler:, errorHandler:)
```

Interactive Messaging

Replying

Optional handler

Recommended

```
func sendMessage(message:, replyHandler:, errorHandler:)

func sendMessageData(data:, replyHandler:, errorHandler:)
```

Interactive Messaging

Replying

Optional handler

Recommended

- Confirmation by receiver

```
func sendMessage(message:, replyHandler:, errorHandler:)

func sendMessageData(data:, replyHandler:, errorHandler:)
```

Interactive Messaging

Replying

Optional handler

Recommended

- Confirmation by receiver

Separate delegate callbacks

```
func sendMessage(message:, replyHandler:, errorHandler:)

func sendMessageData(data:, replyHandler:, errorHandler:)
```

Interactive Messaging

Delegate callbacks

Interactive Messaging

Delegate callbacks

```
func session(session: WCSession, didReceiveMessage message: [String :
AnyObject], replyHandler: ([String : AnyObject]) -> Void) {
    // Handle message, return reply
}
```

Interactive Messaging

Delegate callbacks

```
func session(session: WCSession, didReceiveMessage message: [String :
AnyObject], replyHandler: ([String : AnyObject]) -> Void) {
    // Handle message, return reply
}
```

```
func session(session: WCSession, didReceiveMessage message: [String :
AnyObject]) {
    // Handle message
}
```

Interactive Messaging

Code

Interactive Messaging

Code

```
if (WCSession.defaultSession().reachable) {
```

Interactive Messaging

Code

```
if (WCSession.defaultSession().reachable) {
    let message = // Create dictionary of content
```

Interactive Messaging

Code

```
if (WCSession.defaultSession().reachable) {
    let message = // Create dictionary of content
    WCSession.defaultSession().sendMessage(message,
```

Interactive Messaging

Code

```
if (WCSession.defaultSession().reachable) {
    let message = // Create dictionary of content
    WCSession.defaultSession().sendMessage(message,
        replyHandler: { ([String : AnyObject]) -> Void in
            // Handle reply
    })
```

Interactive Messaging

Code

```
if (WCSession.defaultSession().reachable) {
    let message = // Create dictionary of content
    WCSession.defaultSession().sendMessage(message,
        replyHandler: { ([String : AnyObject]) -> Void in
            // Handle reply
        })
         errorHandler: { (NSError) -> Void in
            // Handle error
    });
}
```

WatchConnectivity

WatchConnectivity

Setup

WatchConnectivity

Setup

- Delegate and activate

WatchConnectivity

Setup

- Delegate and activate

Session state

WatchConnectivity

Setup

- Delegate and activate

Session state

Background transfers

WatchConnectivity

Setup

- Delegate and activate

Session state

Background transfers

- Application context

WatchConnectivity

Setup

- Delegate and activate

Session state

Background transfers

- Application context
- User info transfer

WatchConnectivity

Setup

- Delegate and activate

Session state

Background transfers

- Application context
- User info transfer
- File transfer

WatchConnectivity

Setup

- Delegate and activate

Session state

Background transfers

- Application context
- User info transfer
- File transfer

Interactive messaging

WatchConnectivity

Setup

- Delegate and activate

Session state

Background transfers

- Application context
- User info transfer
- File transfer

Interactive messaging

- Live communication

NSURLSession

NSURLSession

Introduction

NSURLSession

Introduction

Existing foundation class

NSURLSession

Introduction

Existing foundation class

HTTP requests

NSURLSession

Introduction

Existing foundation class

HTTP requests

Available in watchOS 2

NSURLSession

Introduction

Existing foundation class

HTTP requests

Available in watchOS 2

Tetherless Wi-Fi

NSURLSession

When to use

NSURLSession

When to use

Server has new content

NSURLSession

When to use

Server has new content

Similar to iOS apps

NSURLSession

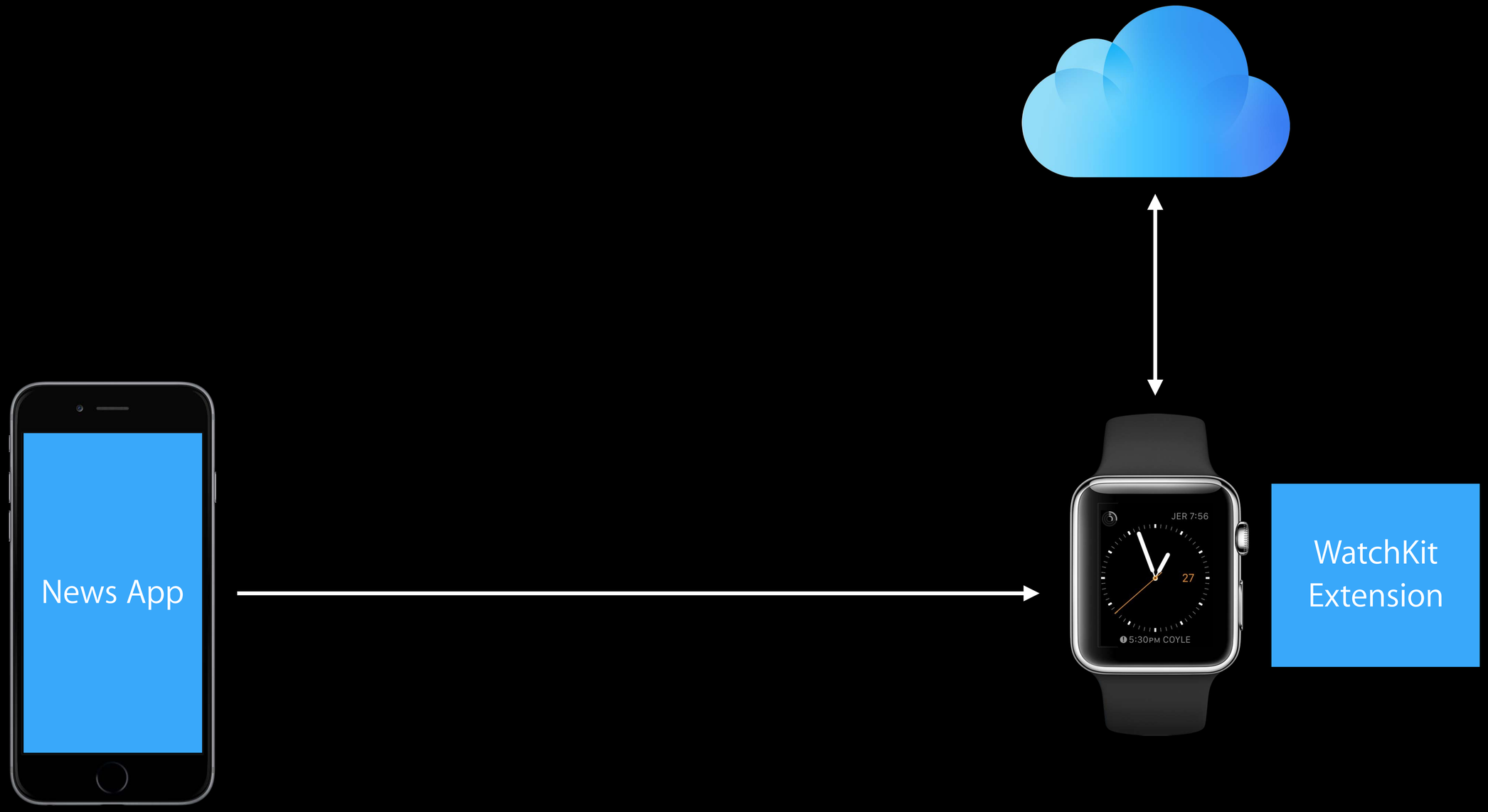
When to use

Server has new content

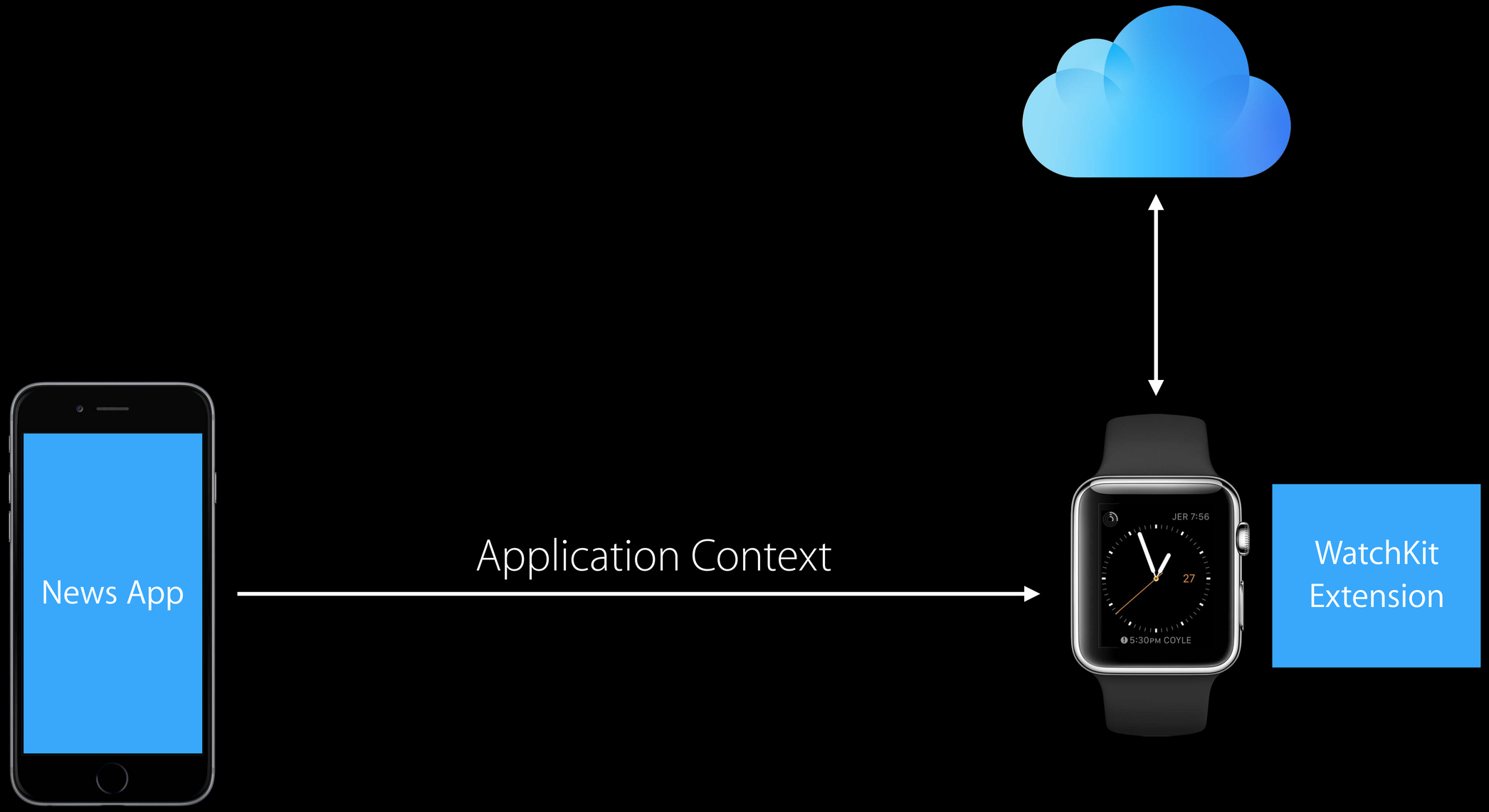
Similar to iOS apps

- Content tailored for Apple Watch

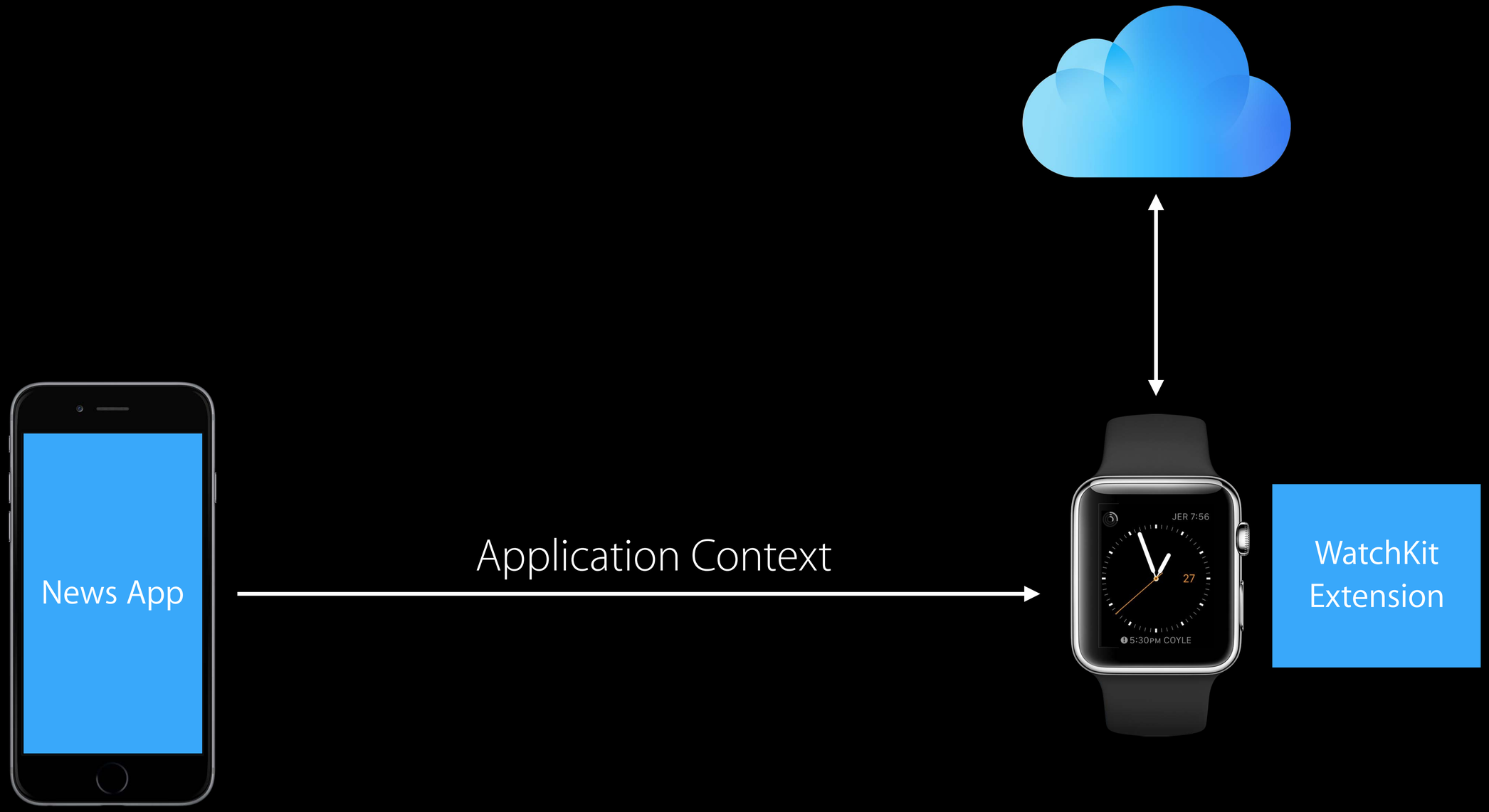
NSURLSession and WatchConnectivity



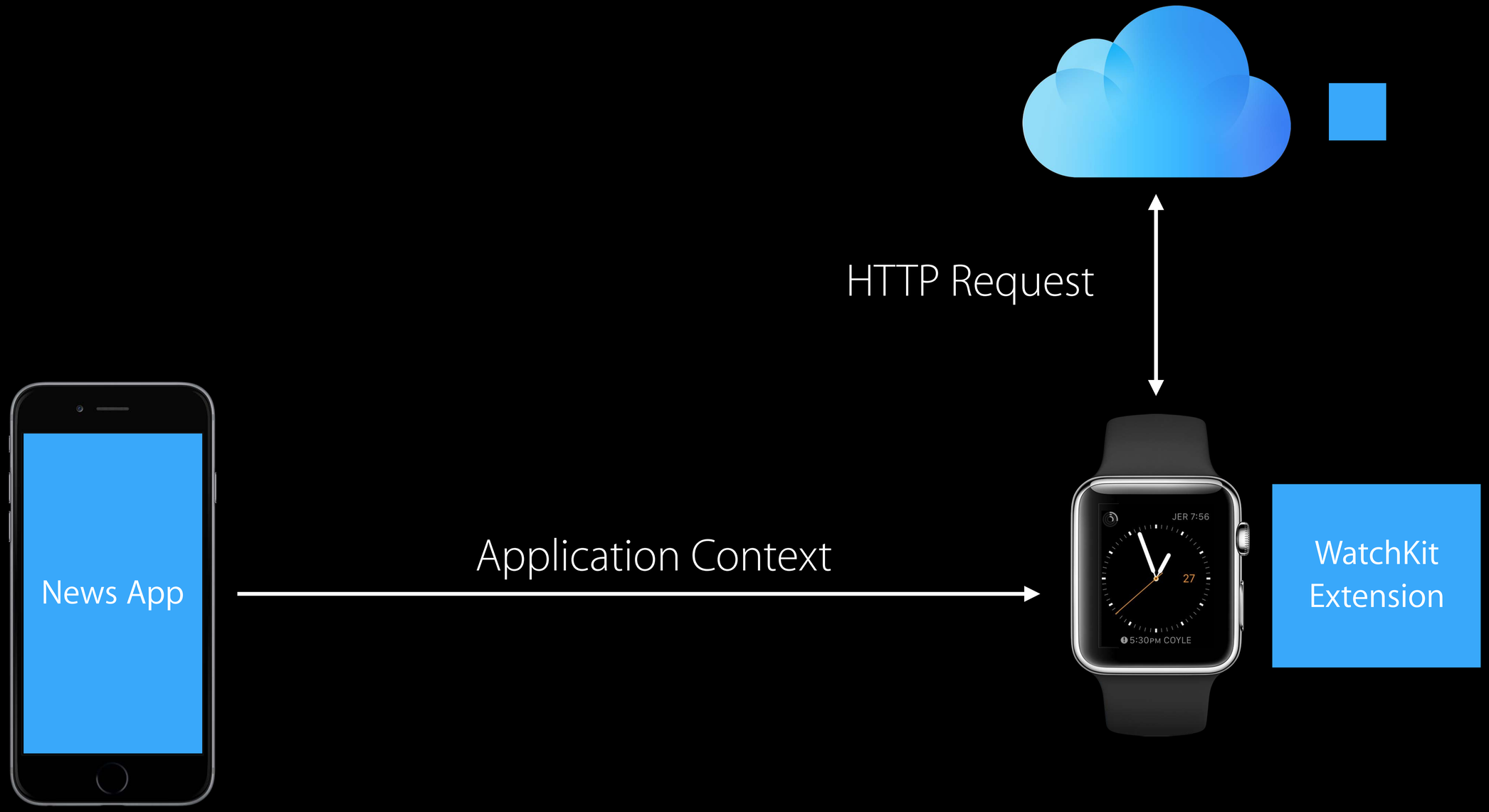
NSURLSession and WatchConnectivity



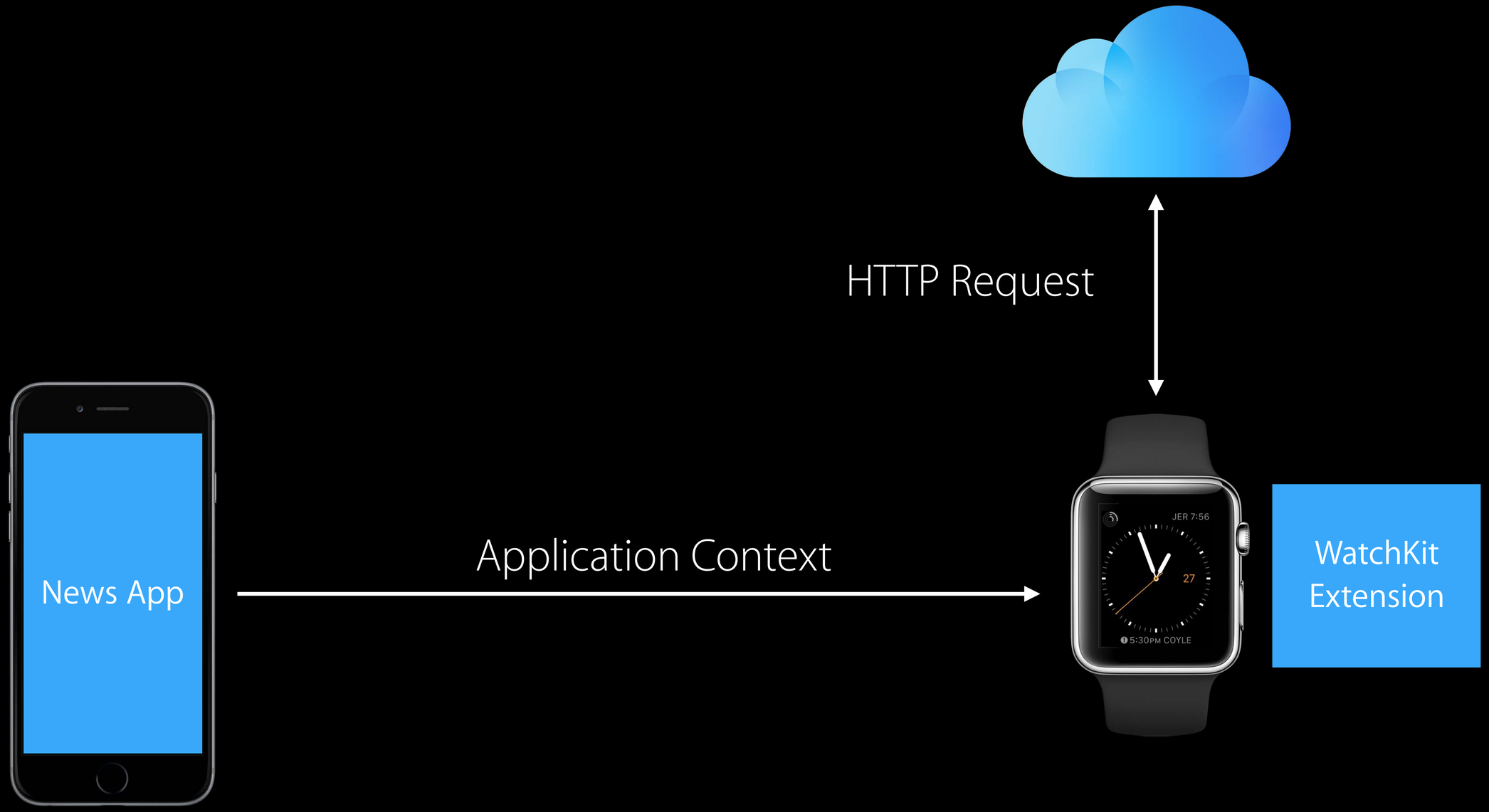
NSURLSession and WatchConnectivity



NSURLSession and WatchConnectivity



NSURLSession and WatchConnectivity



Complications

Complications

Complications

Complications

Introduction

Complications

Introduction

Two primary tasks

Complications

Introduction

Two primary tasks

- Updating clock face

Complications

Introduction

Two primary tasks

- Updating clock face
- Get content to Apple Watch

Complications

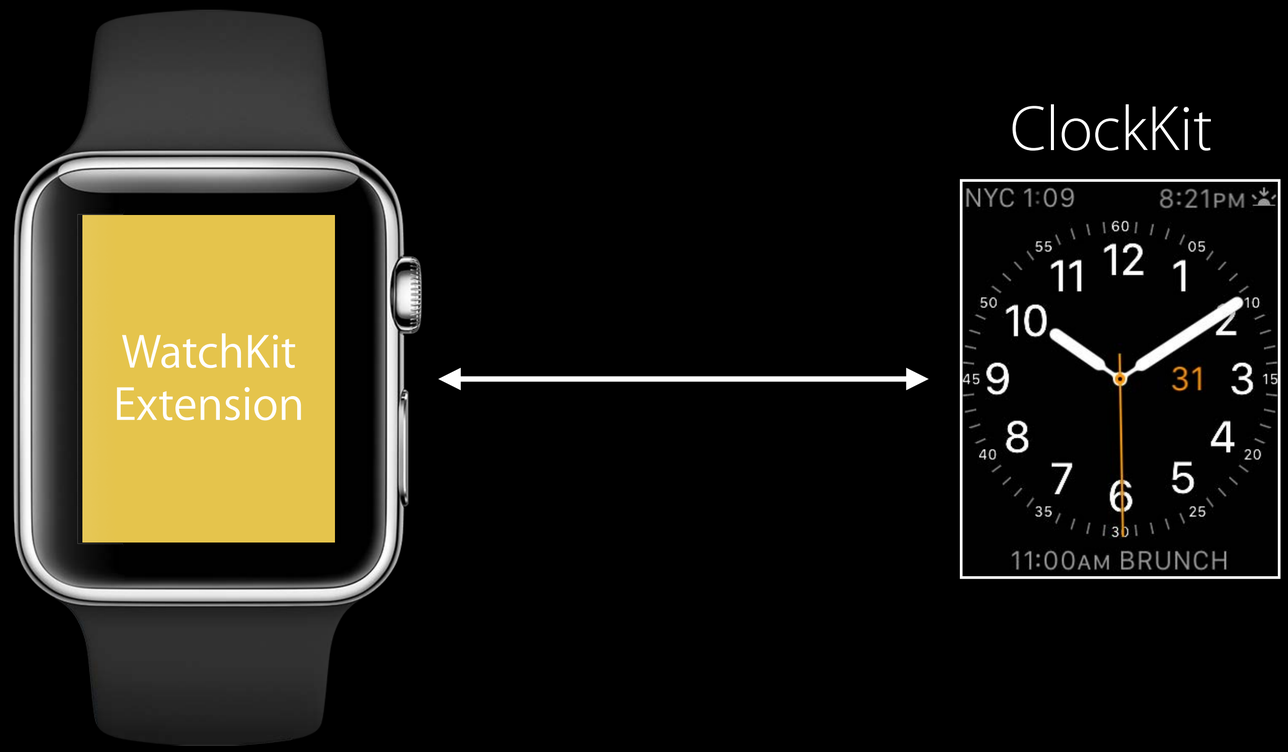
Updating clock face

Complications

Updating clock face

Complications

Updating clock face



Complications

Updating clock face

ClockKit

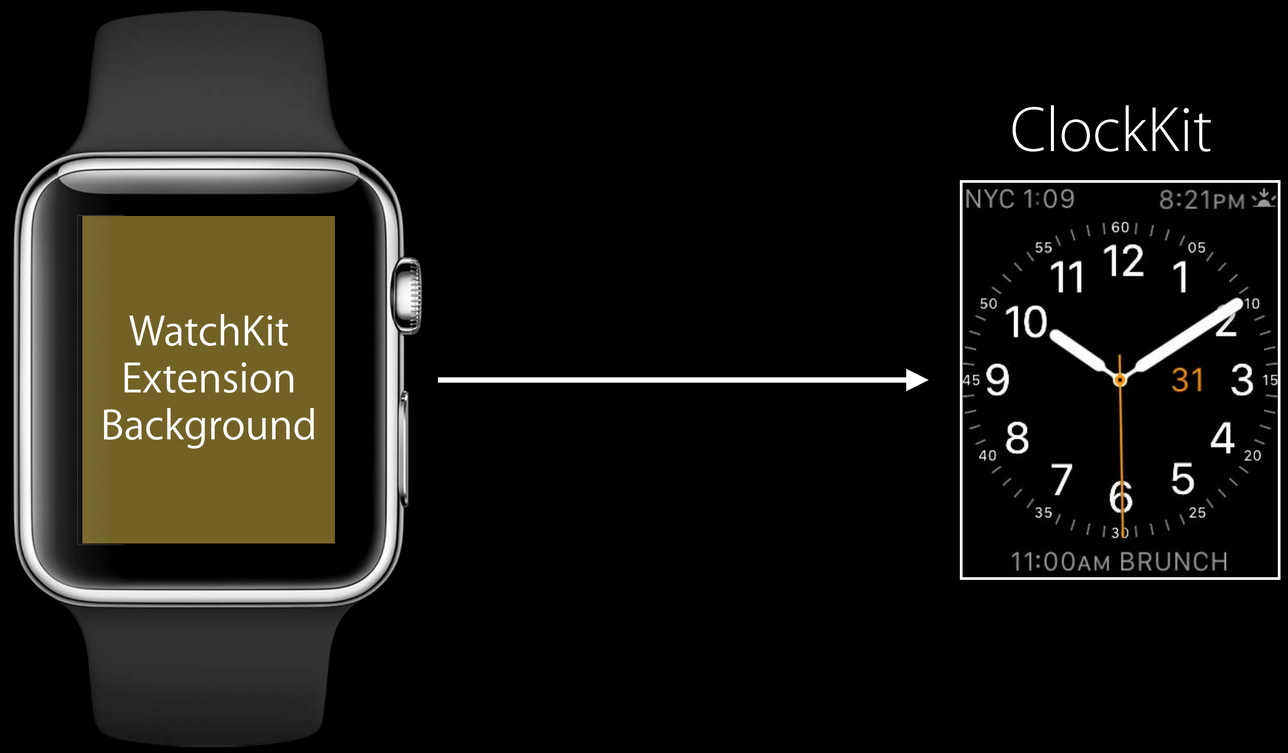
Complications

Updating clock face

ClockKit

Complications

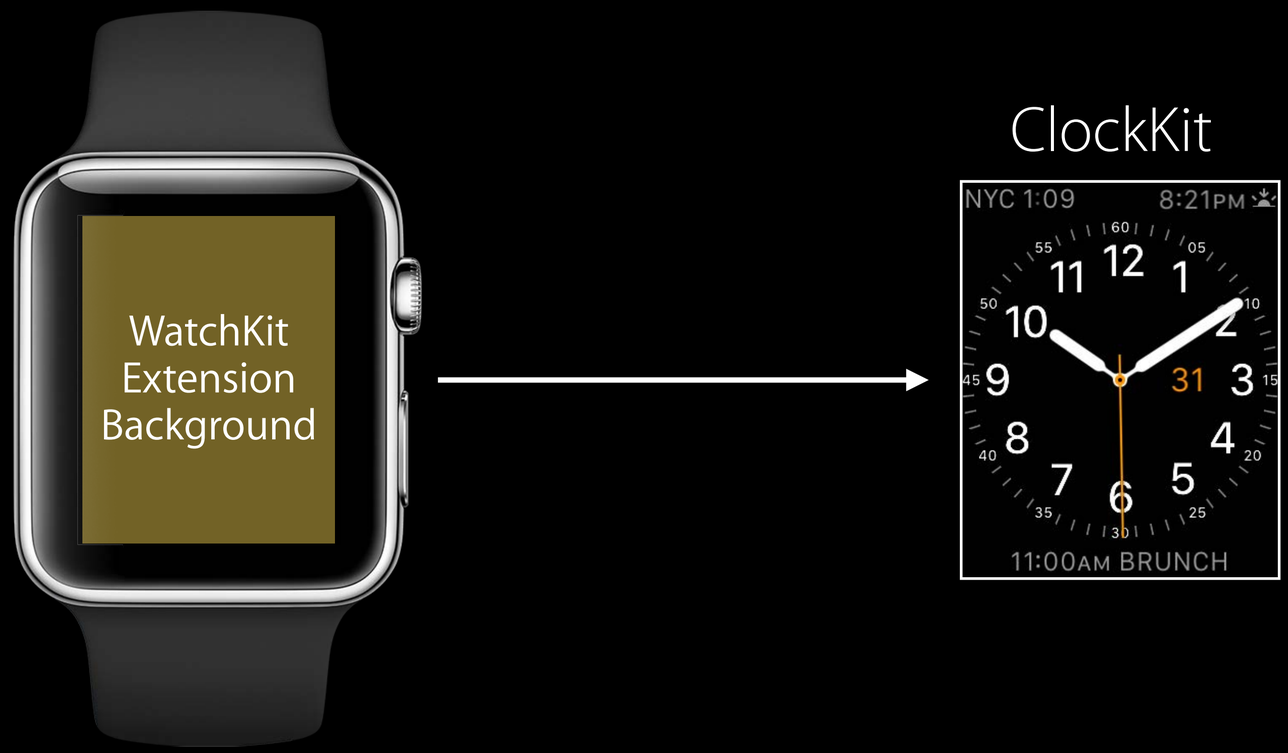
Updating clock face




```
let server = CLKComplicationServer.sharedInstance()
```

Complications

Updating clock face




```
let server = CLKComplicationServer.sharedInstance()
server.extendTimelineForComplication(aComplication)
```

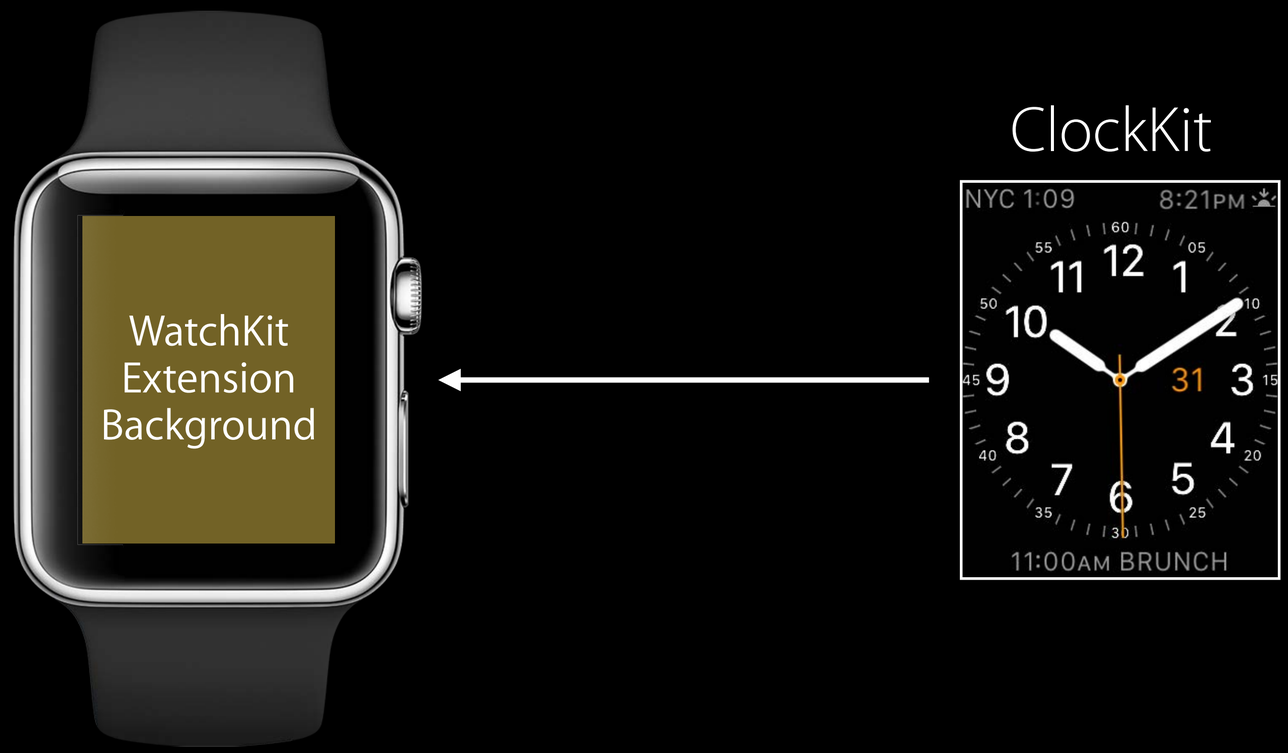
Complications

Updating clock face

ClockKit

Complications

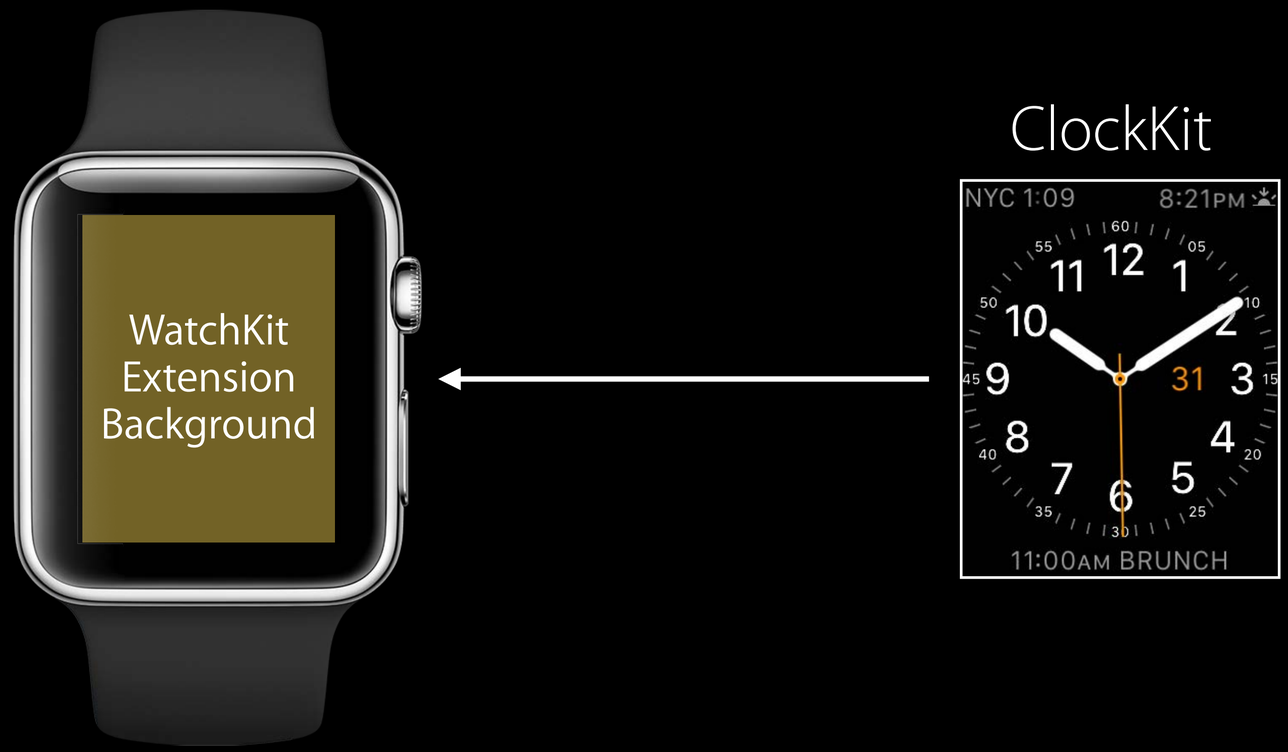
Updating clock face




```
getCurrentTimelineEntryForComplication(_:withHandler:)
```

Complications

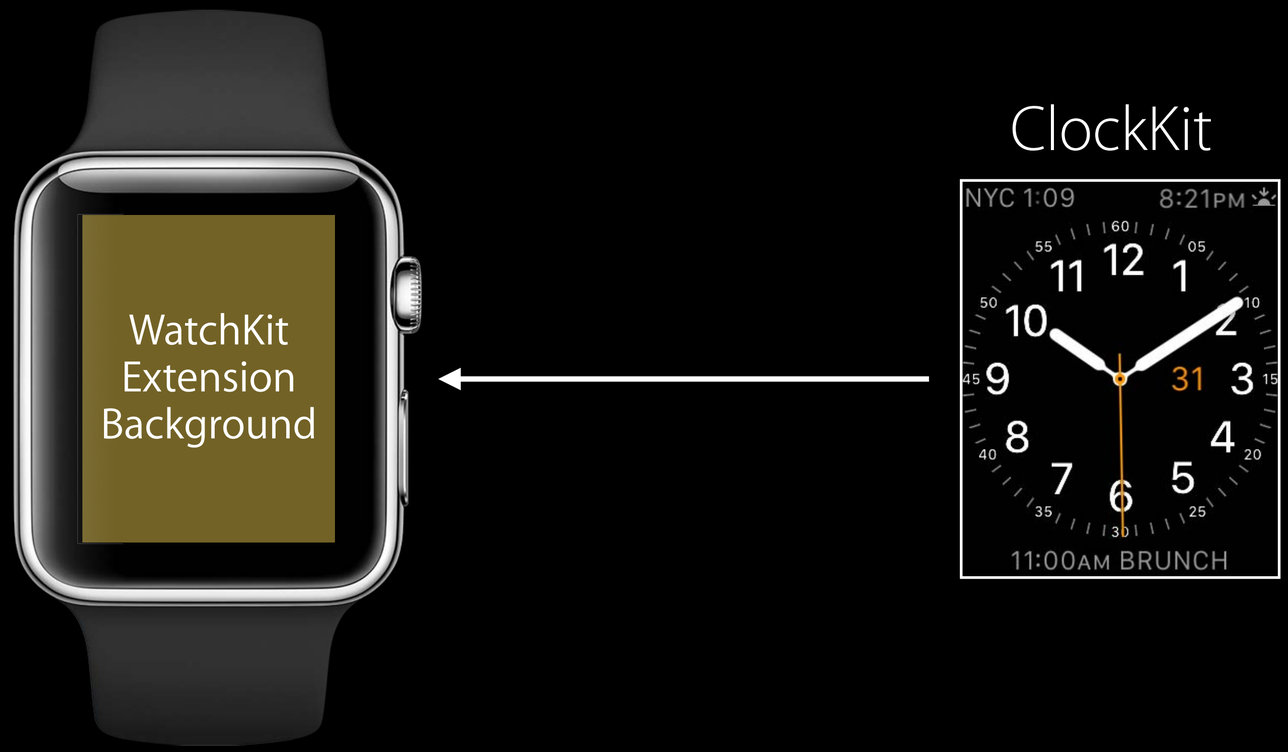
Updating clock face




```
getCurrentTimelineEntryForComplication(_:withHandler:)
getTimelineEntriesForComplication(_:beforeDate:limit:withHandler:)
```

Complications

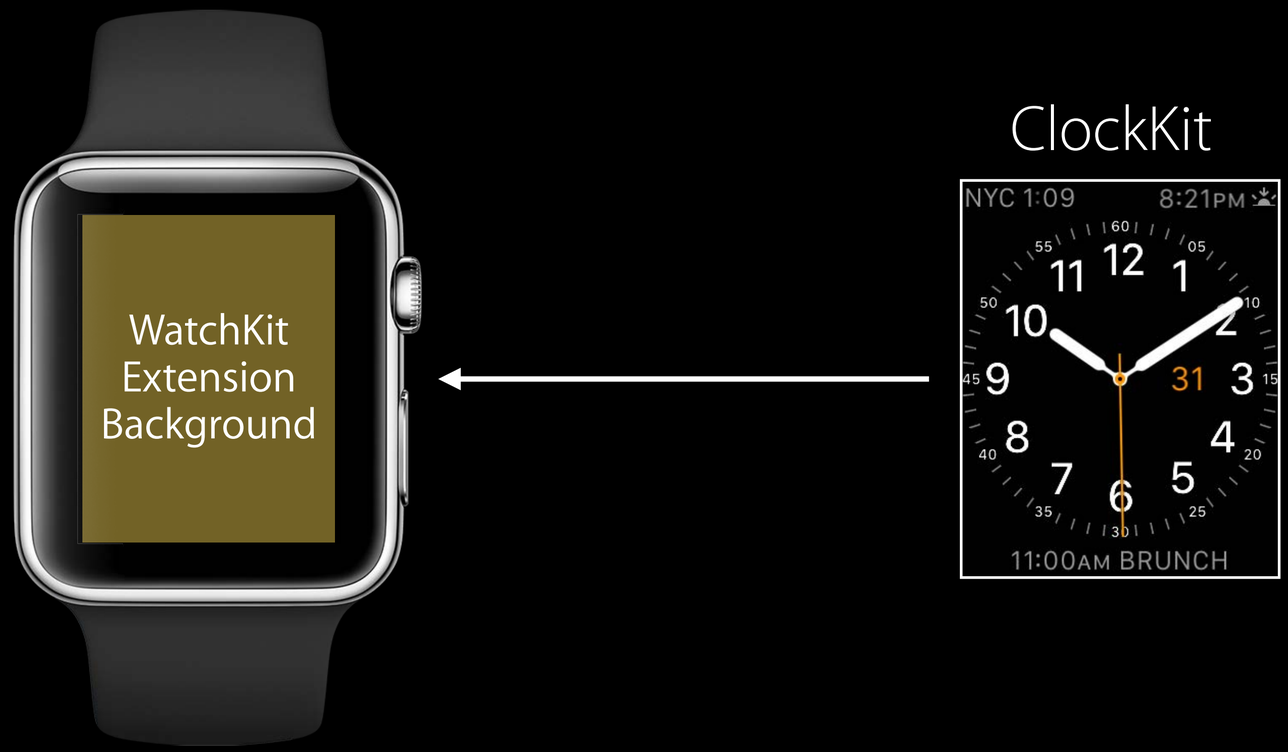
Updating clock face




```
getCurrentTimelineEntryForComplication(_:withHandler:)
getTimelineEntriesForComplication(_:beforeDate:limit:withHandler:)
getTimelineEntriesForComplication(_:afterDate:limit:withHandler:)
```

Complications

Updating clock face




```
getCurrentTimelineEntryForComplication(_:withHandler:)
getTimelineEntriesForComplication(_:beforeDate:limit:withHandler:)
getTimelineEntriesForComplication(_:afterDate:limit:withHandler:)
getNextRequestedUpdateDateWithHandler(_)
```

Complications

Updating clock face

Complications

Updating clock face

ClockKit

Complications

Updating clock face

ClockKit

Content for past, present, and future

Complications

Updating clock face

ClockKit

Content for past, present, and future

Background launched for updating

Complications

Updating clock face

ClockKit

Content for past, present, and future

Background launched for updating

Specify when content provided will be stale

Complications

Updating clock face

ClockKit

Content for past, present, and future

Background launched for updating

Specify when content provided will be stale **BUDGETED**

Complications

Updating clock face

ClockKit

Content for past, present, and future

Background launched for updating

Specify when content provided will be stale **BUDGETED**

Initial Activation

Complications

Complications

Initial activation

Complications

Initial activation

Complications

Initial activation

Complications

Initial activation

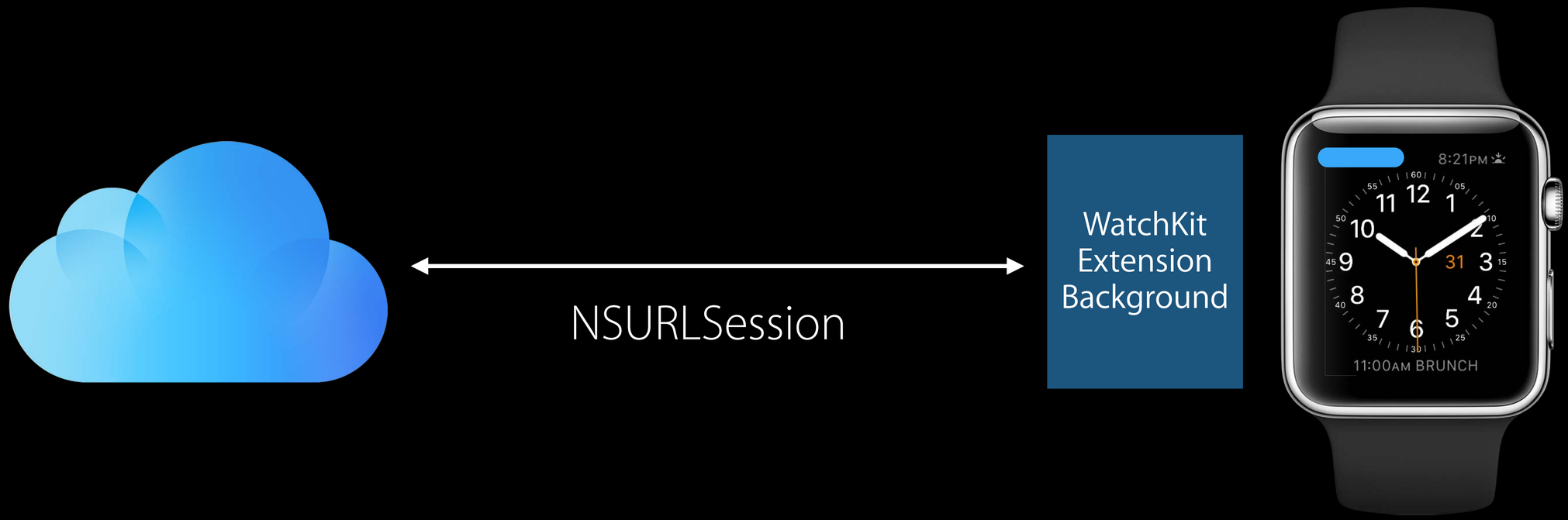
Complications

Initial activation

WatchKit Extension Background

Complications

Initial activation



Complications

Initial activation

WatchKit Extension Background

Complications

Initial activation

WatchKit Extension Background

Complications

Initial activation

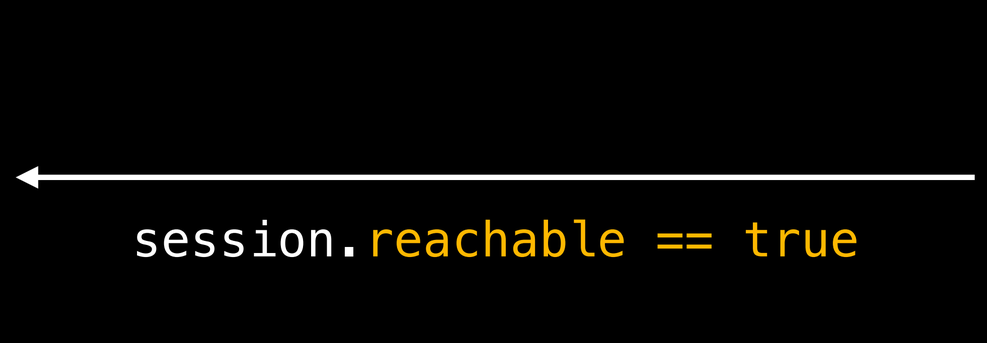
session.reachable == true

WatchKit Extension Background

Complications

Initial activation

`sendMessage(_:replyHandler:errorHandler:)`



`session.reachable == true`

WatchKit Extension Background

Complications

Initial activation

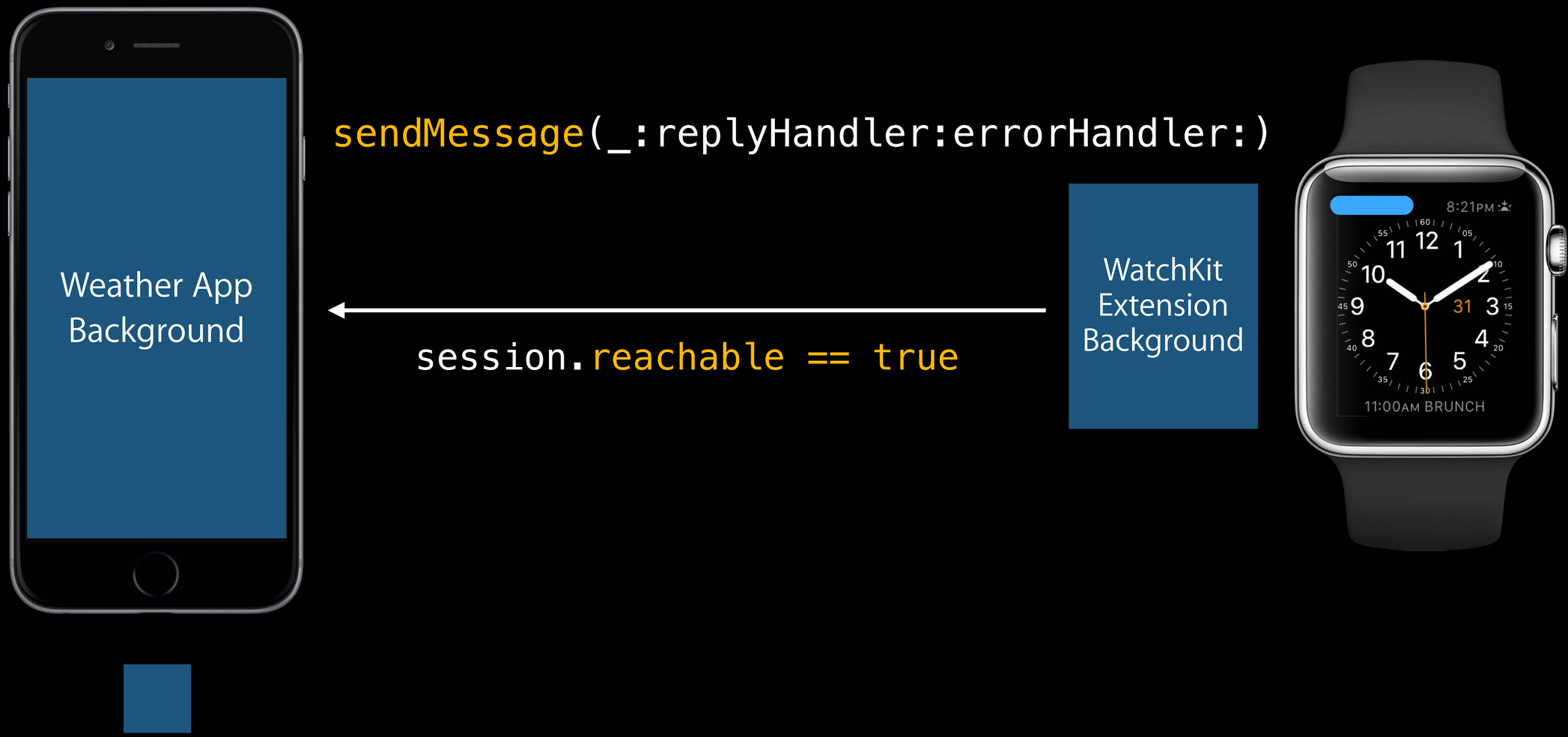
`sendMessage(_:replyHandler:errorHandler:)`

`session.reachable == true`

WatchKit Extension Background

Complications

Initial activation



Complications

Initial activation

Complications

Initial activation: Summary

Complications

Initial activation: Summary

WatchKit extension will get launched in the background

Complications

Initial activation: Summary

WatchKit extension will get launched in the background

NSURLSession

Complications

Initial activation: Summary

WatchKit extension will get launched in the background

NSURLSession

Special complication update where iOS app is `reachable`

Complications

Initial activation: Summary

WatchKit extension will get launched in the background

NSURLSession

Special complication update where iOS app is `reachable`

Populate as much as possible of the ClockKit timeline

Staying Current

Complications

Complications

Staying current

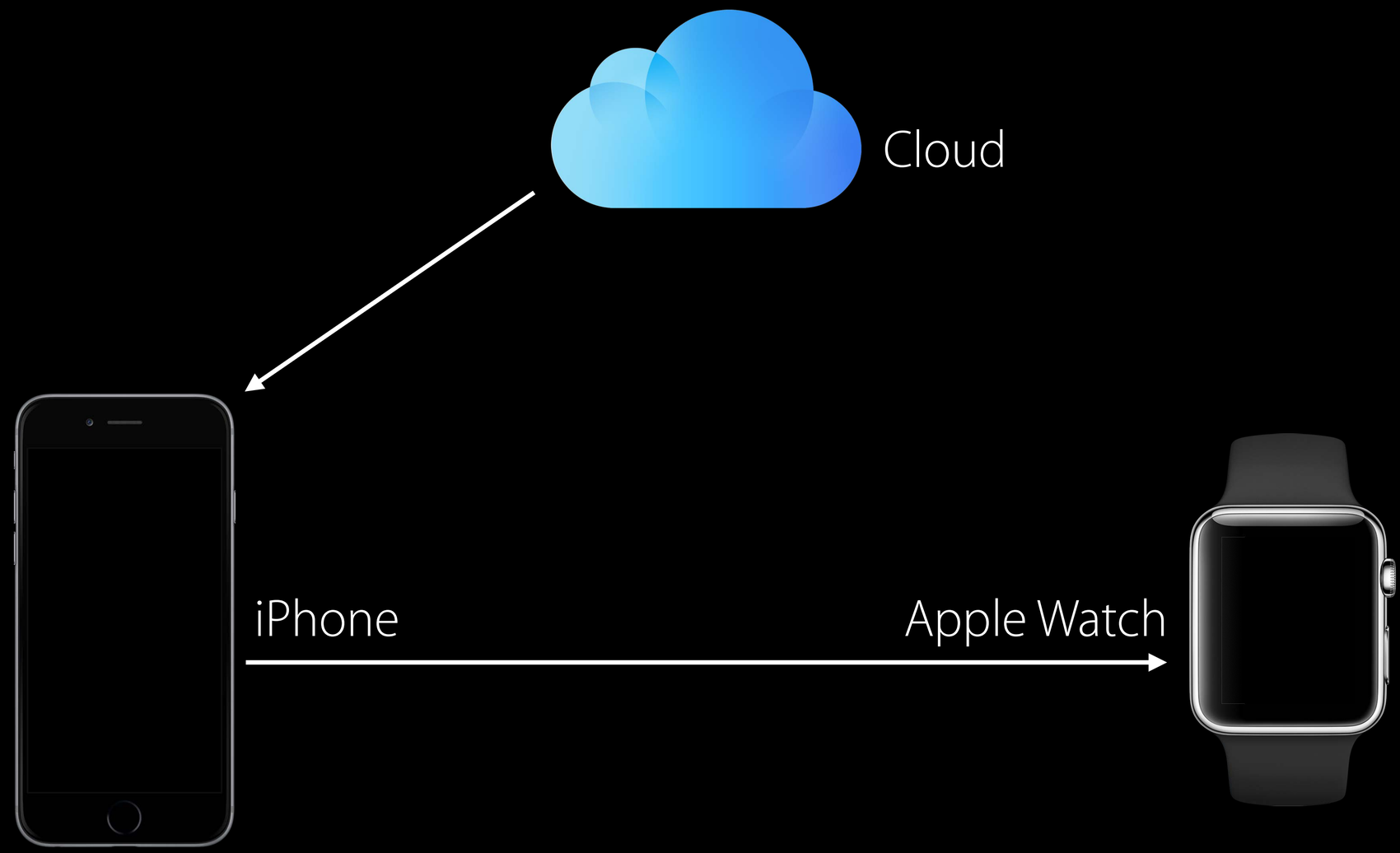
Cloud

iPhone

Apple Watch

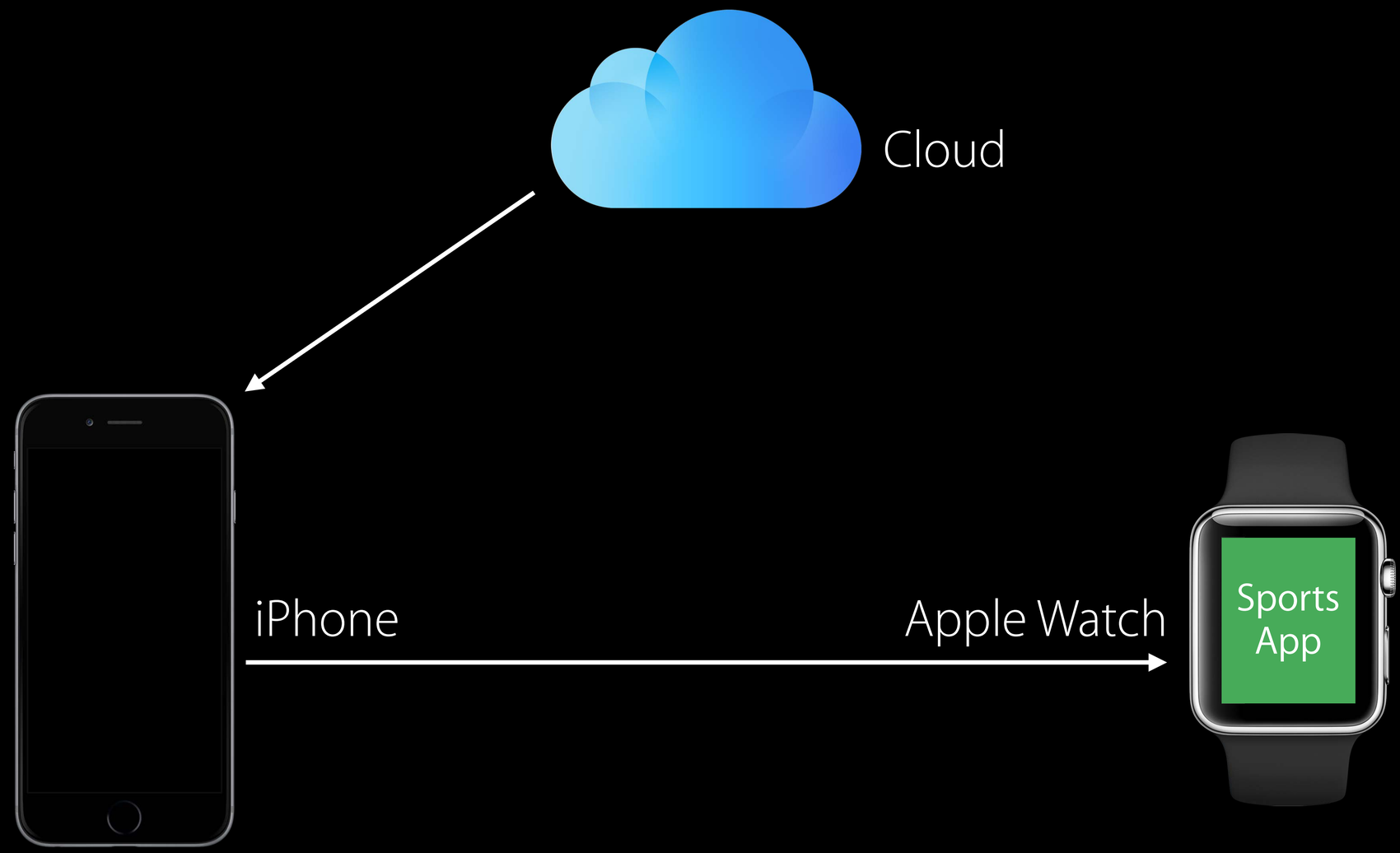
Complications

Pushed



Complications

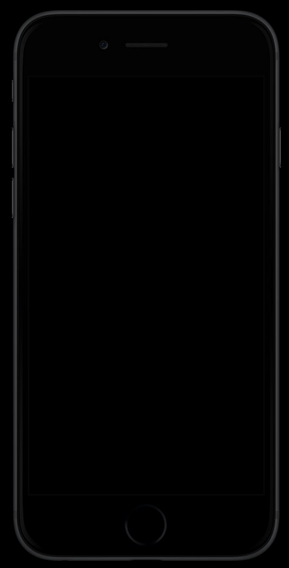
Pushed



Complications

Requested interval fetch

Cloud



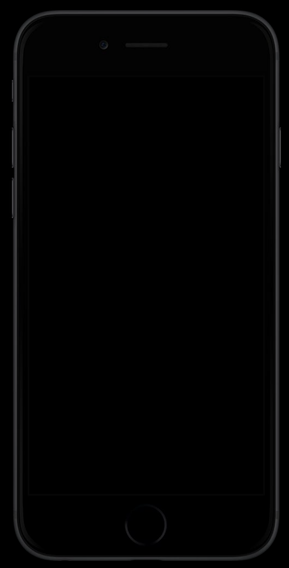
iPhone

Apple Watch

Complications

Requested interval fetch

Cloud



iPhone

Apple Watch

Complications

Requested interval fetch

NSURLSession

Complications

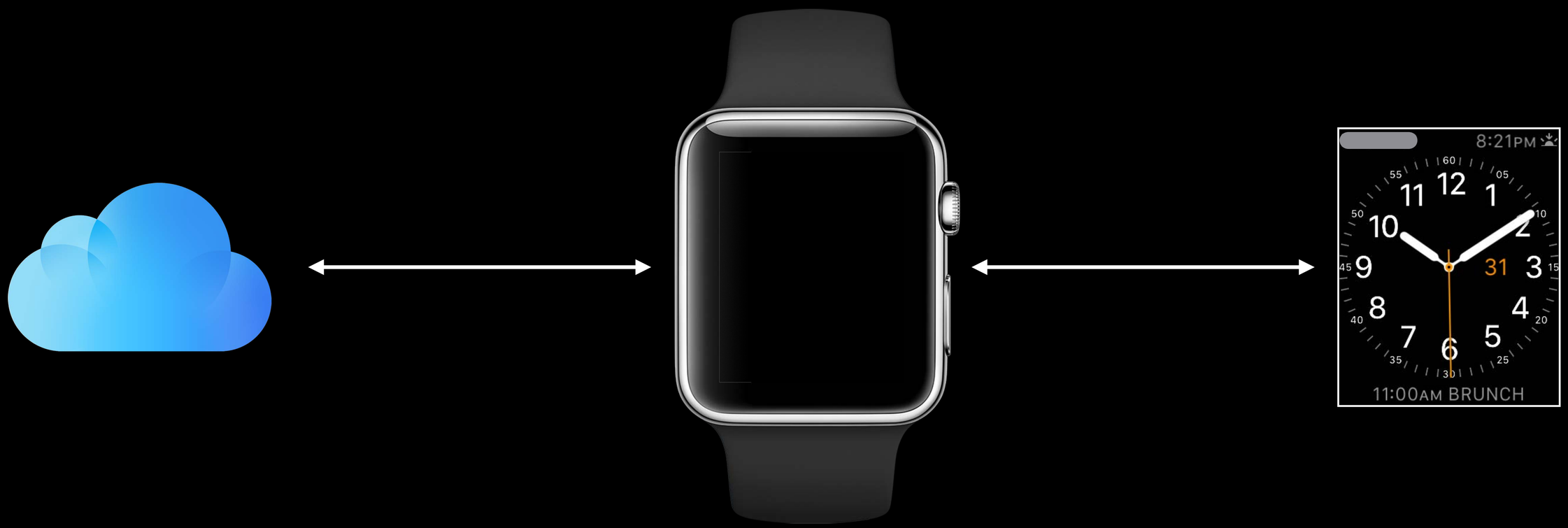
Requested interval fetch

NSURLSession

ClockKit

Complications

Requested interval fetch



Complications

Requested interval fetch

Complications

Requested interval fetch

Complications

Requested interval fetch

Complications

Requested interval fetch

Complications

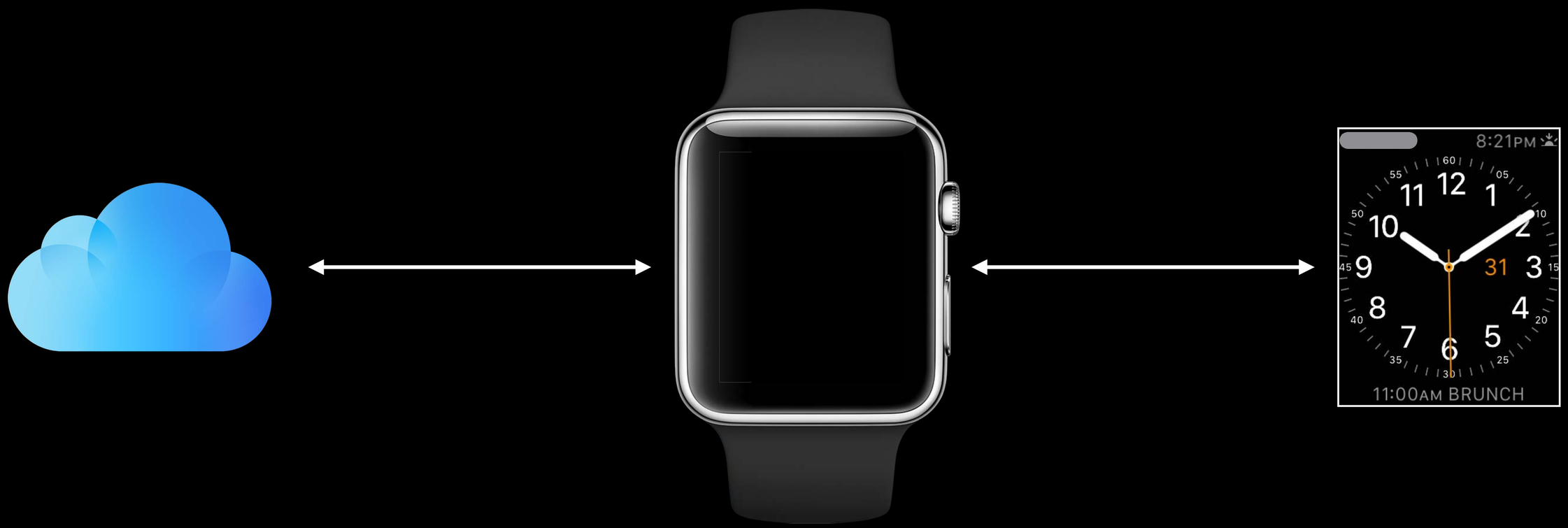
Requested interval fetch

Complications

Requested interval fetch

Complications

Requested interval fetch



Complications

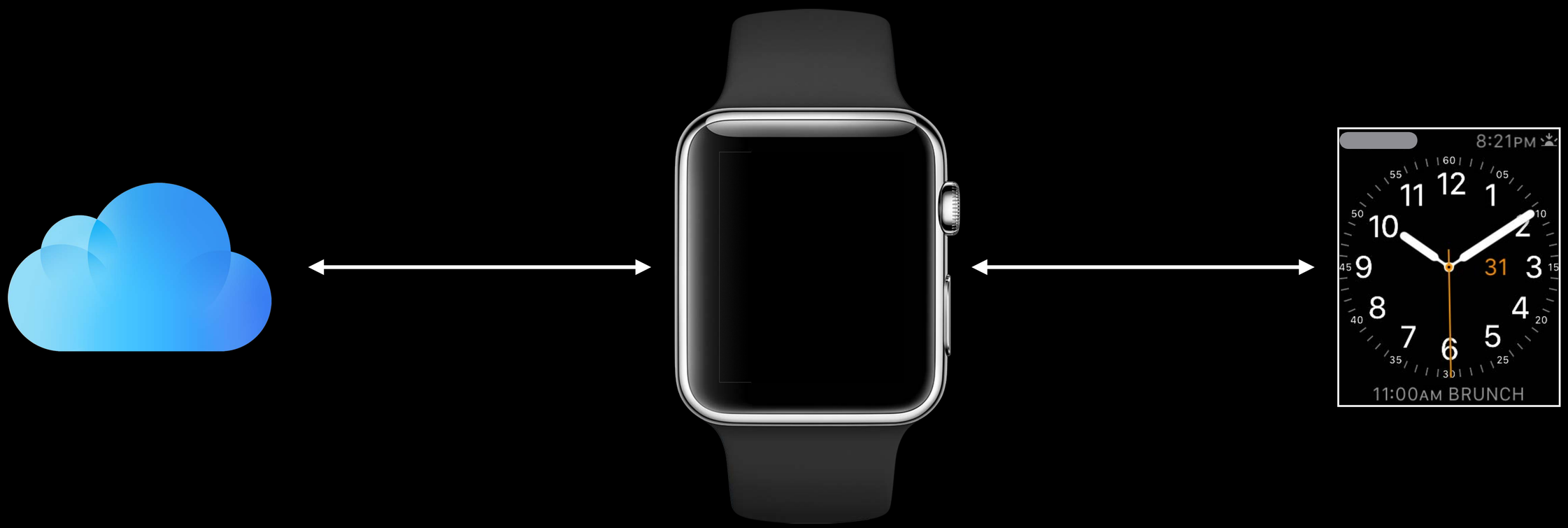
Requested interval fetch

Complications

Requested interval fetch

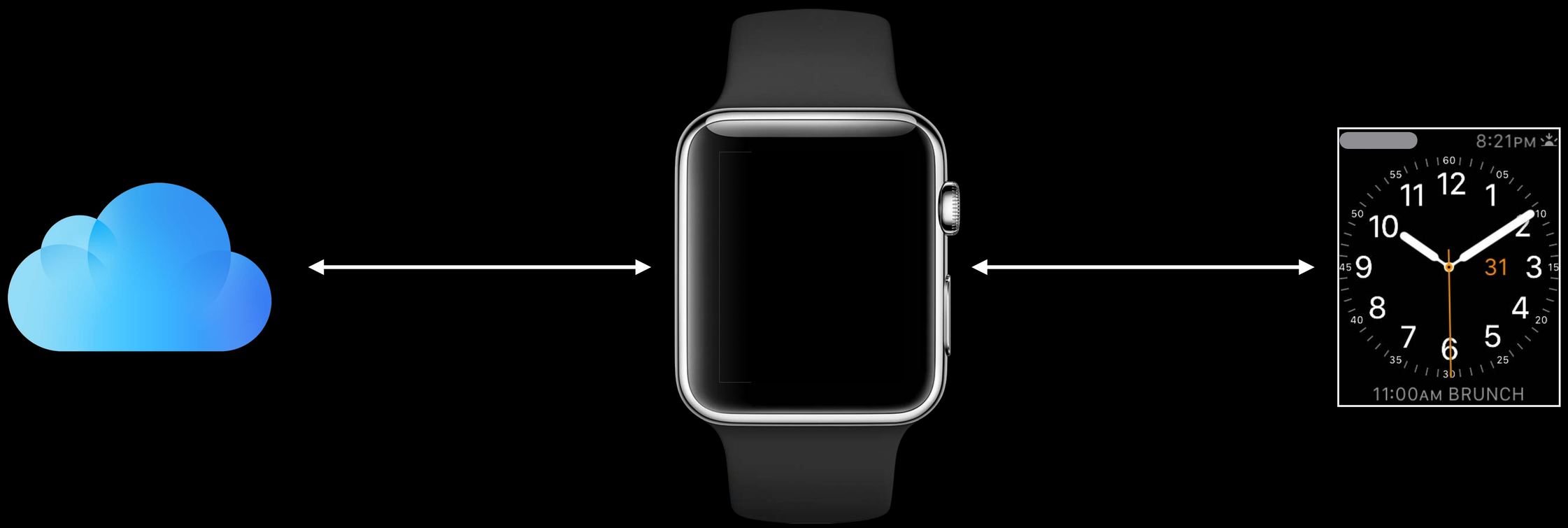
Complications

Requested interval fetch



Complications

Requested interval fetch



Complications

Requested interval fetch: Summary

Complications

Requested interval fetch: Summary

Use a NSURLSession background session if possible

Complications

Requested interval fetch: Summary

Use a NSURLSession background session if possible

NSURLSession requests might not complete until next time the extension runs

Complications

Requested interval fetch: Summary

Use a NSURLSession background session if possible

NSURLSession requests might not complete until next time the extension runs

Requested update time is a suggestion

Complications

Requested interval fetch: Summary

Use a NSURLSession background session if possible

NSURLSession requests might not complete until next time the extension runs

Requested update time is a suggestion

Keep runtime as short as possible

Complications

Requested interval fetch: Summary

Use a NSURLSession background session if possible

NSURLSession requests might not complete until next time the extension runs

Requested update time is a suggestion

Keep runtime as short as possible

ClockKit

Complications

Requested interval fetch: Summary

Use a NSURLSession background session if possible

NSURLSession requests might not complete until next time the extension runs

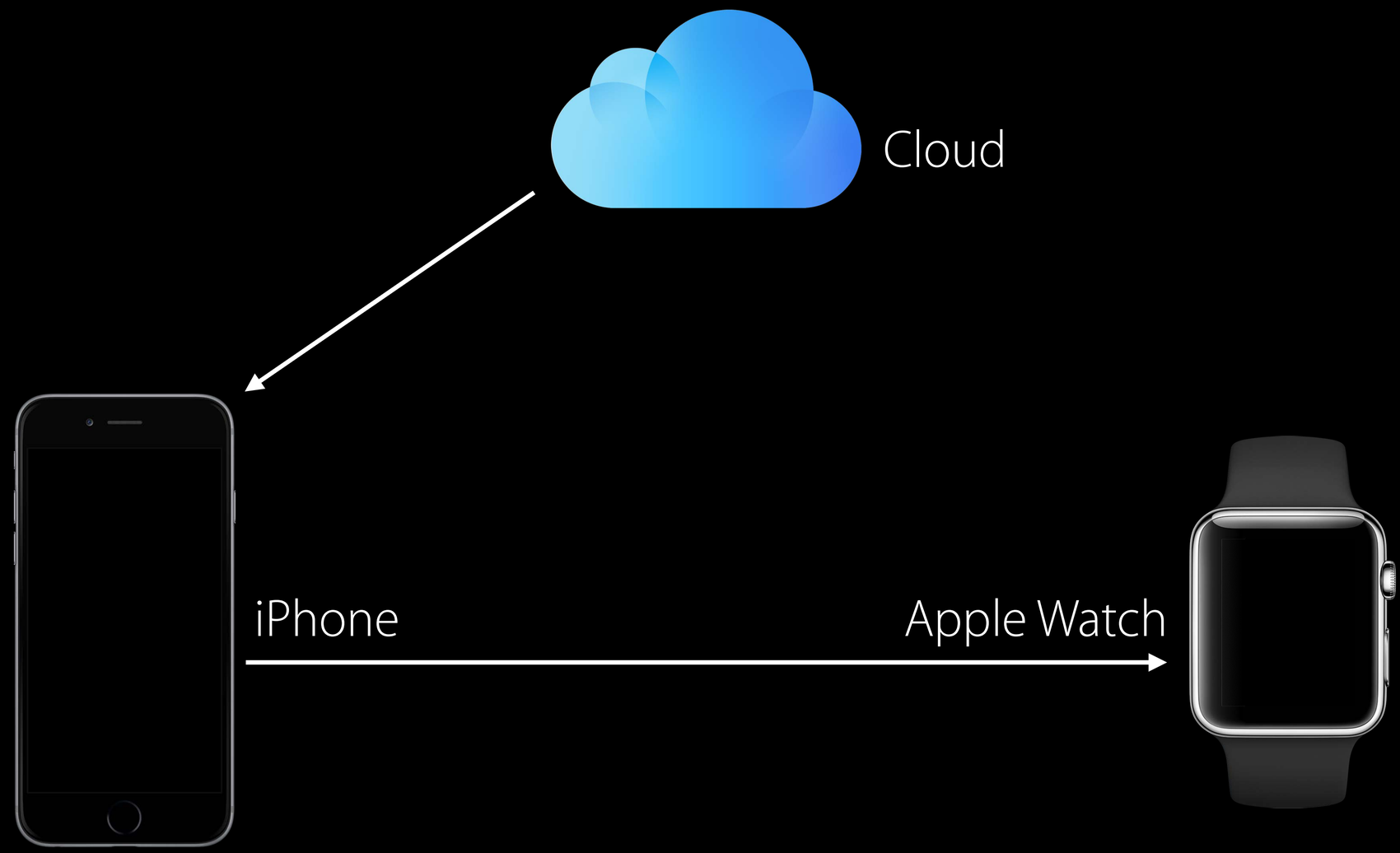
Requested update time is a suggestion **BUDGETED**

Keep runtime as short as possible **BUDGETED**

ClockKit

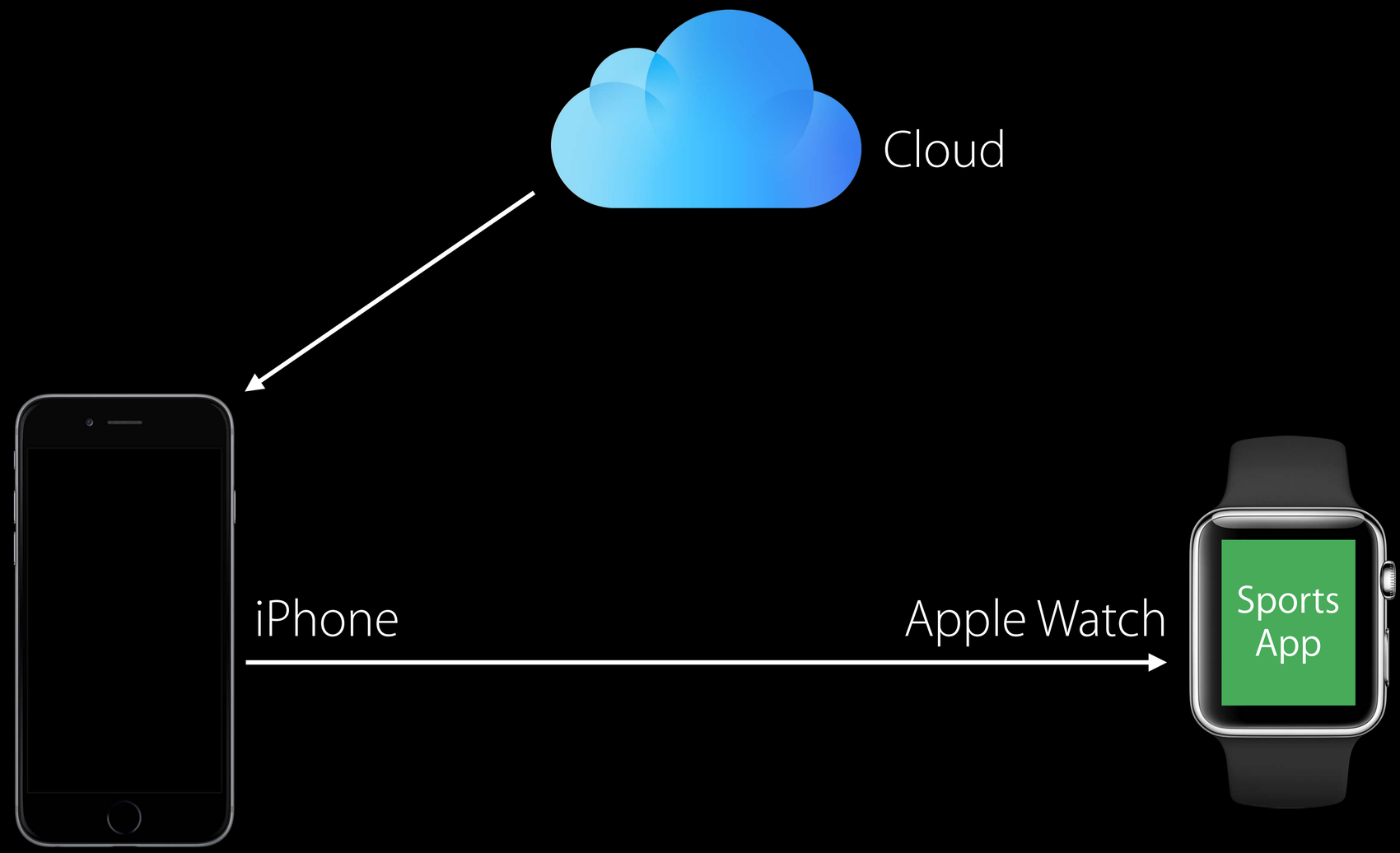
Complications

Pushed



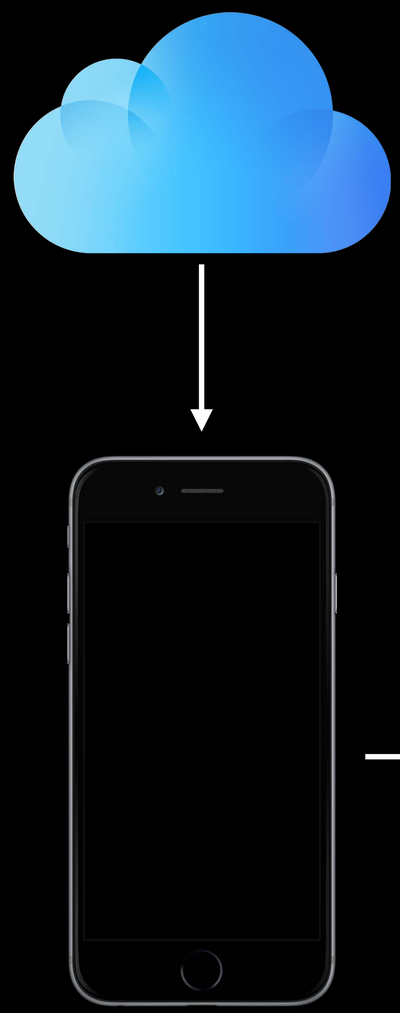
Complications

Pushed



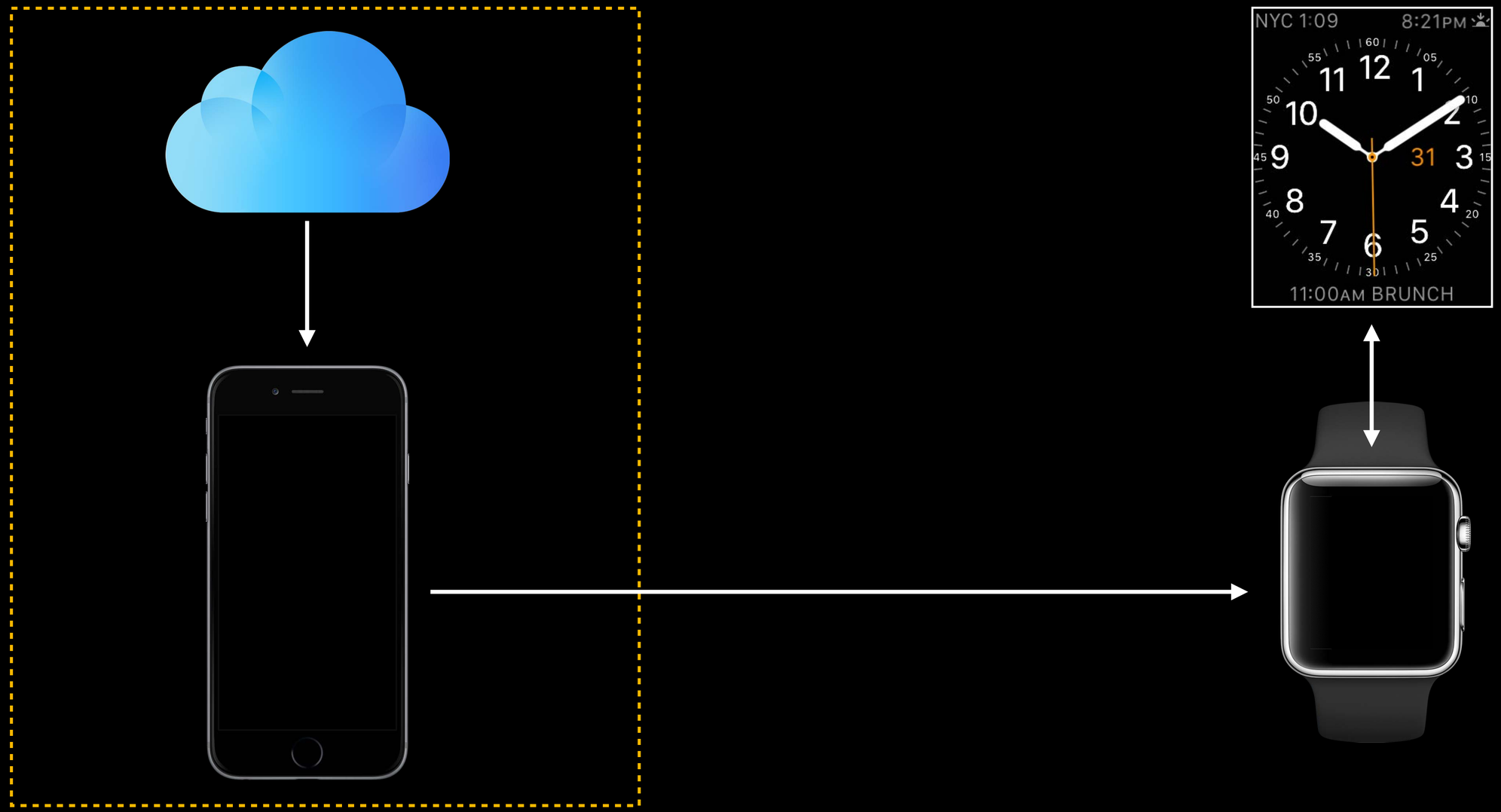
Complications

Pushed



Complications

Pushed: Part 1



Complications

Pushed: Part 1

Complications

Pushed: Part 2

WatchConnectivity

Pushed: Part 1

Complications

Complications

Pushed: Part 1

PushKit

Complications

Pushed: Part 1

Complications

Pushed: Part 1

Updated PushKit framework

Complications

Pushed: Part 1

Updated PushKit framework

```
pushRegistry = PKPushRegistry(queue: dispatch_get_main_queue())
```

Complications

Pushed: Part 1

Updated PushKit framework

```
pushRegistry = PKPushRegistry(queue: dispatch_get_main_queue())
pushRegistry.delegate = self // conforms to PKPushRegistryDelegate
```

Complications

Pushed: Part 1

Updated PushKit framework

```
pushRegistry = PKPushRegistry(queue: dispatch_get_main_queue())
pushRegistry.delegate = self // conforms to PKPushRegistryDelegate
pushRegistry.desiredPushTypes = [PKPushTypeComplication]
```

Complications

Pushed: Part 1

Updated PushKit framework

```
pushRegistry = PKPushRegistry(queue: dispatch_get_main_queue())
pushRegistry.delegate = self // conforms to PKPushRegistryDelegate
pushRegistry.desiredPushTypes = [PKPushTypeComplication]
```

```
func pushRegistry(registry: PKPushRegistry!, didUpdatePushCredentials
credentials: PKPushCredentials!, forType type: String!)
    // Upload push token to server to enable it to send pushes
```

Complications

Pushed: Part 1

Updated PushKit framework

```
pushRegistry = PKPushRegistry(queue: dispatch_get_main_queue())
pushRegistry.delegate = self // conforms to PKPushRegistryDelegate
pushRegistry.desiredPushTypes = [PKPushTypeComplication]
```

```
func pushRegistry(registry: PKPushRegistry!, didUpdatePushCredentials
credentials: PKPushCredentials!, forType type: String!)
    // Upload push token to server to enable it to send pushes
func pushRegistry(registry: PKPushRegistry!,
didReceiveIncomingPushWithPayload payload: PKPushPayload!, forType type:
String!)
    // Massage content and prepare to send it to the complication
```

Pushed: Part 2

Complications

Complications

Pushed: Part 2

Complications

Pushed: Part 2

Complications

Pushed: Part 2

```
session.transferUserInfo(timeLineEntry1)
session.transferUserInfo(timeLineEntry2)
...
```

Complications

Pushed: Part 2

```
session.transferUserInfo(timeLineEntry1)
session.transferUserInfo(timeLineEntry2)
...
session.transferCurrentComplicationUserInfo(presentTimeLineEntry)
```

Complications

Pushed: Part 2

```
session.transferUserInfo(timeLineEntry1)
session.transferUserInfo(timeLineEntry2)
...
session.transferCurrentComplicationUserInfo(presentTimeLineEntry)
```

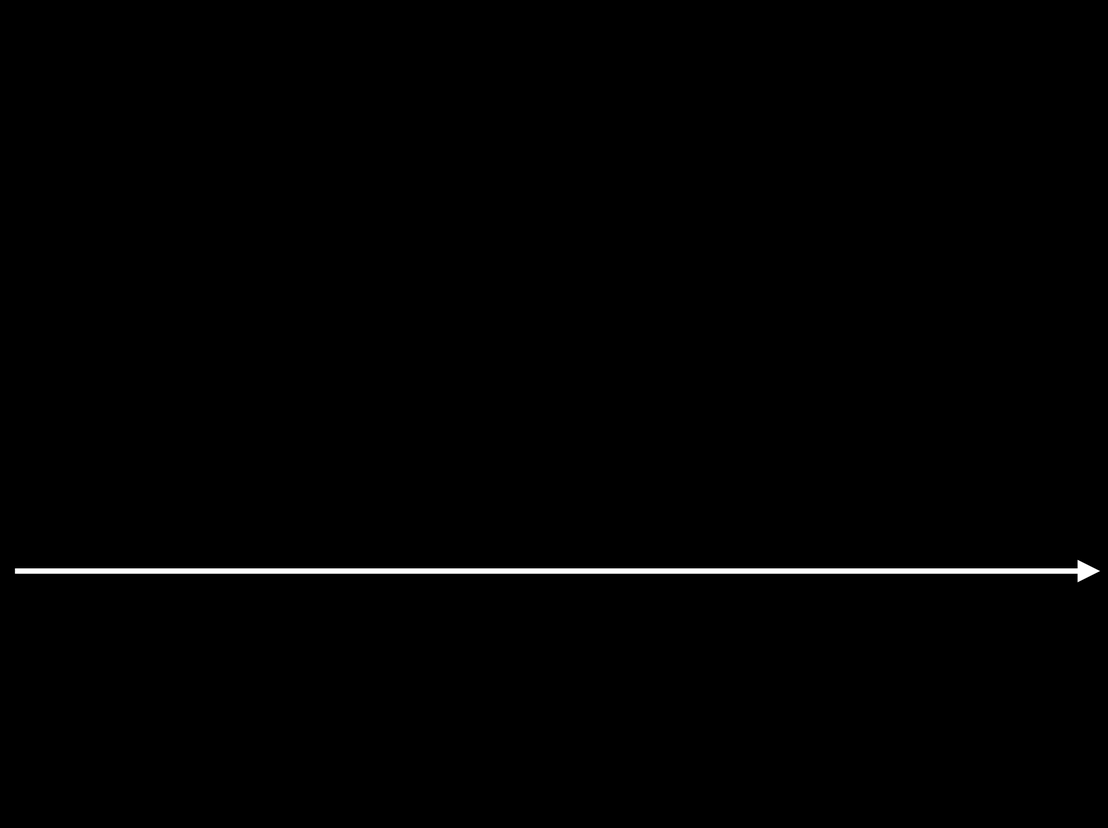
```
func session(session: WCSession, didReceiveUserInfo userInfo: [String :
AnyObject]) {
    // Call ClockKit APIs to update complication
}
```

Pushed: Summary

Complications

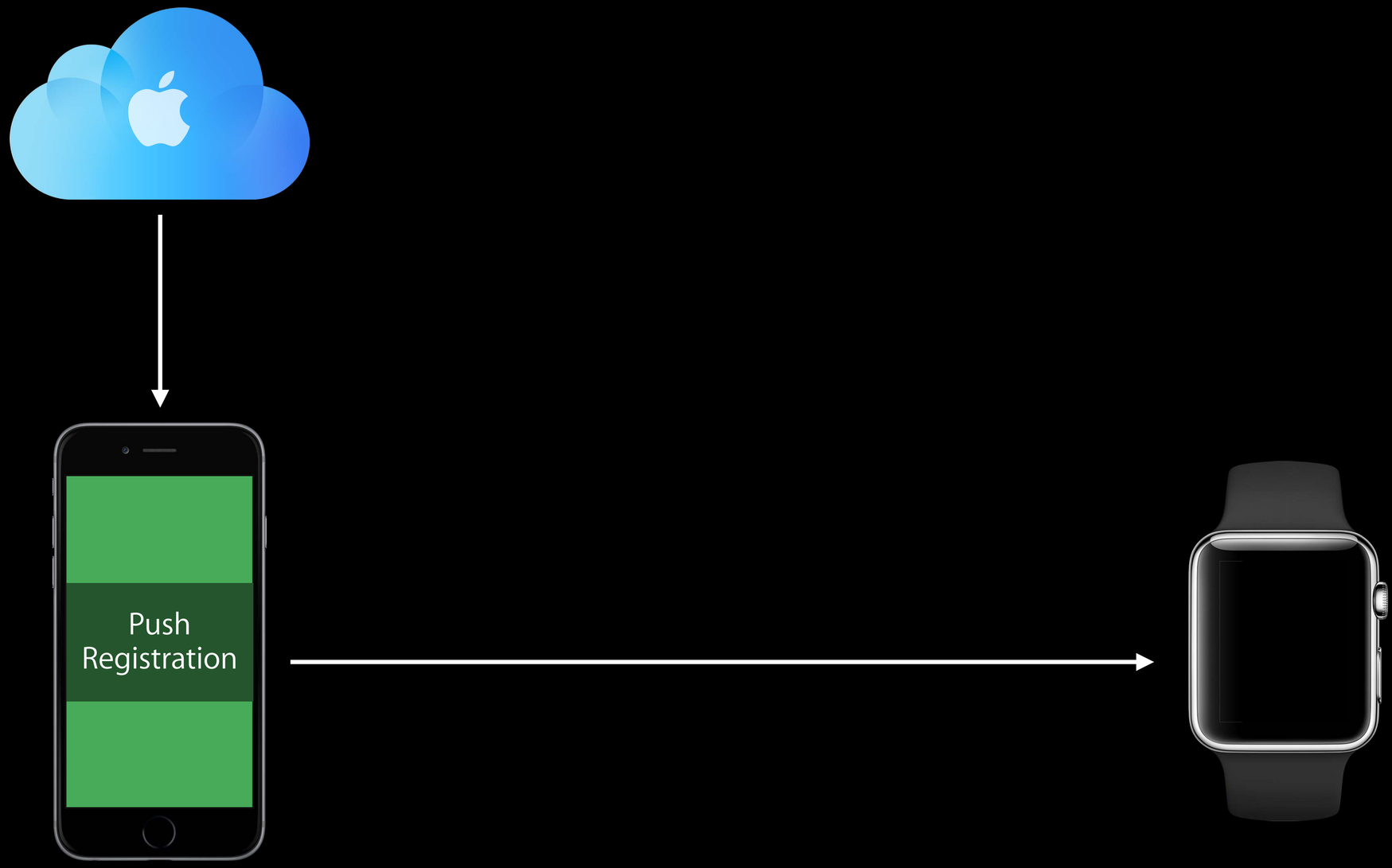
Complications

Pushed: Summary



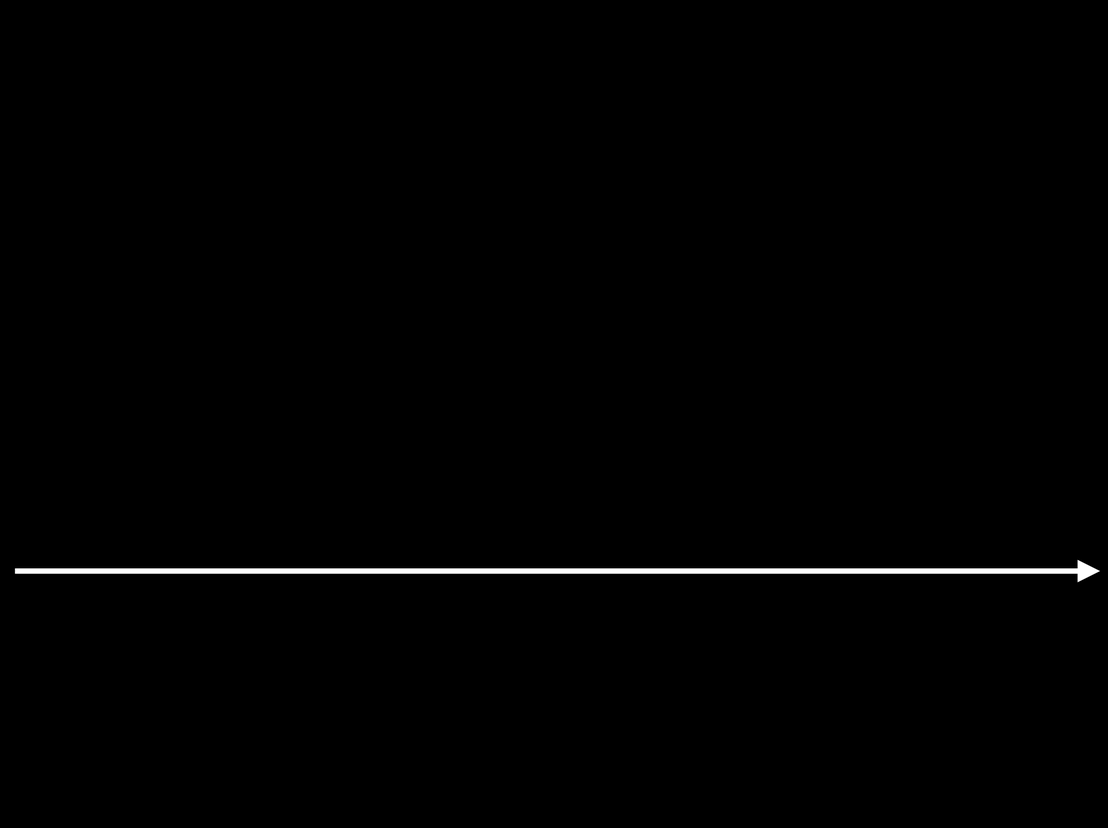
Complications

Pushed: Summary



Complications

Pushed: Summary

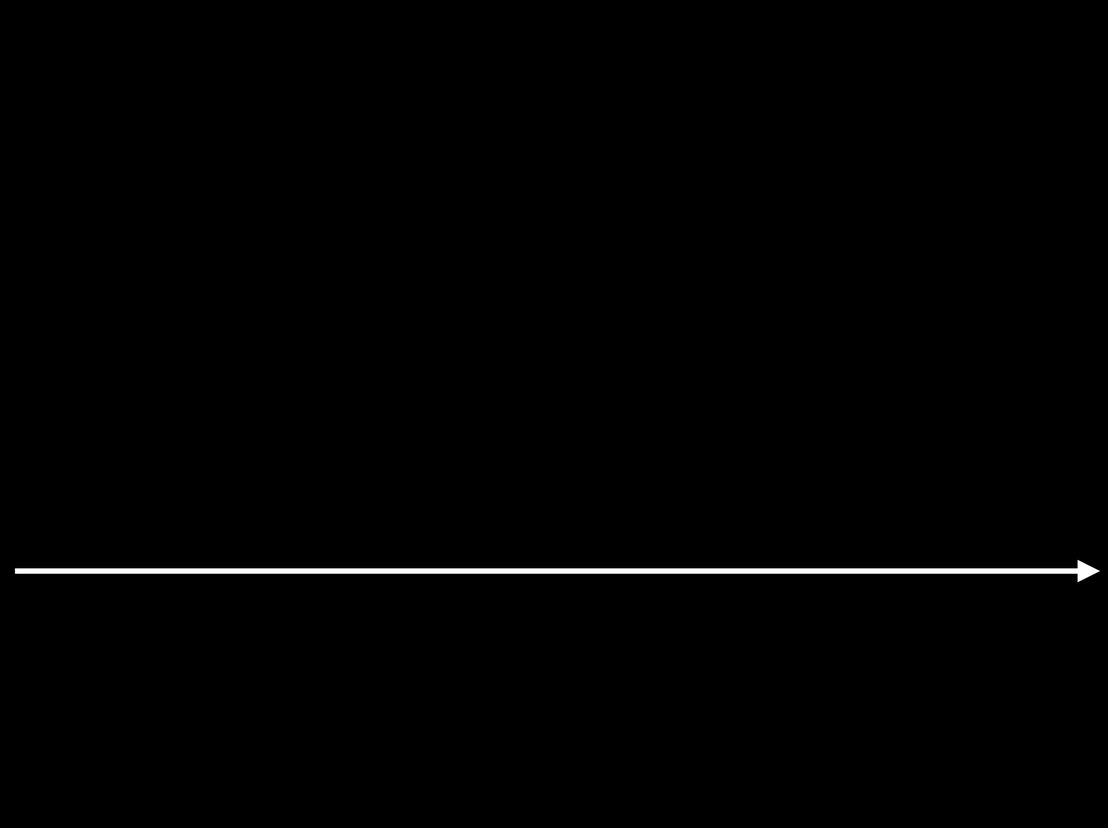


Complications

Pushed: Summary

Complications

Pushed: Summary



Complications

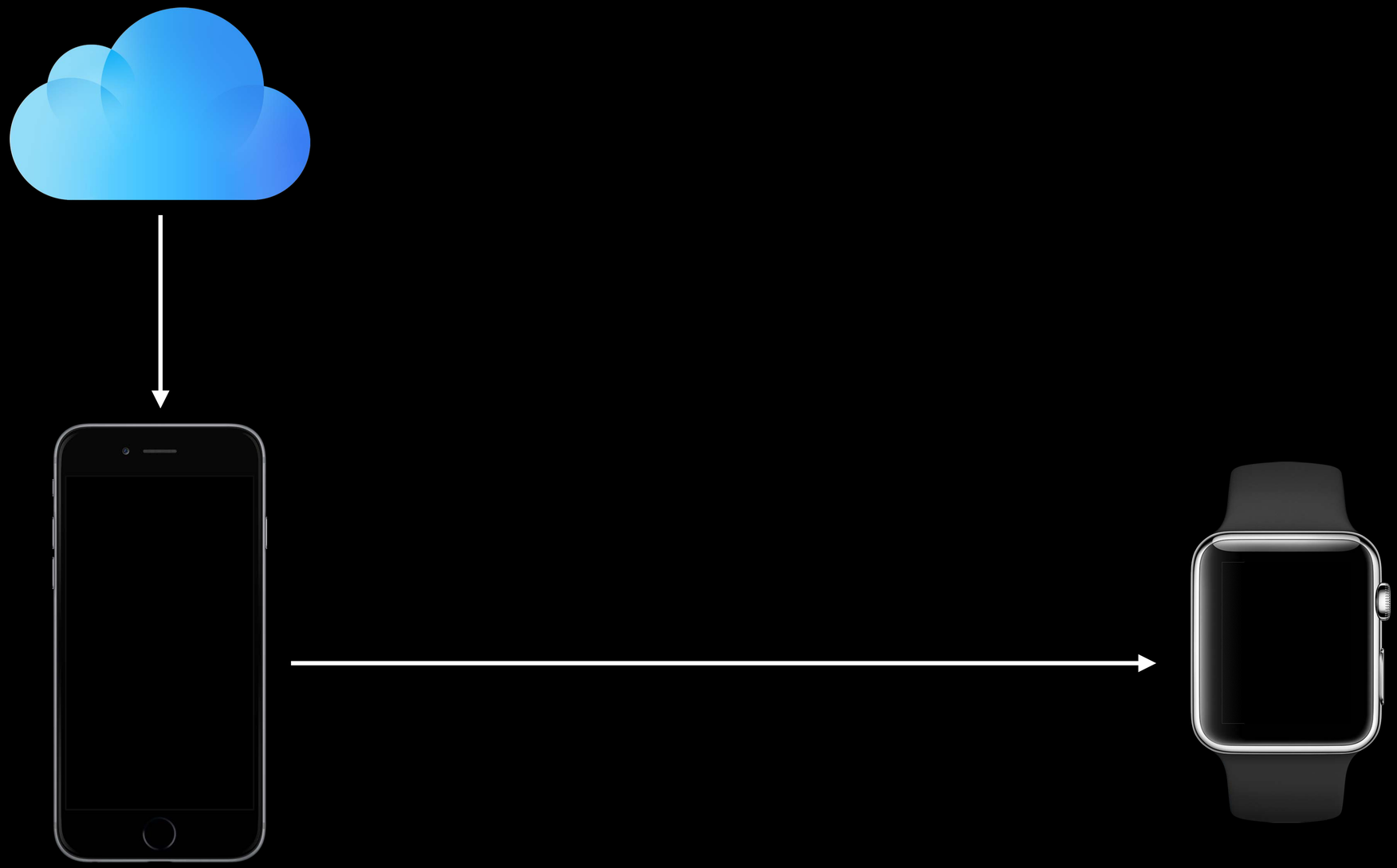
Pushed: Summary

Complications

Pushed: Summary

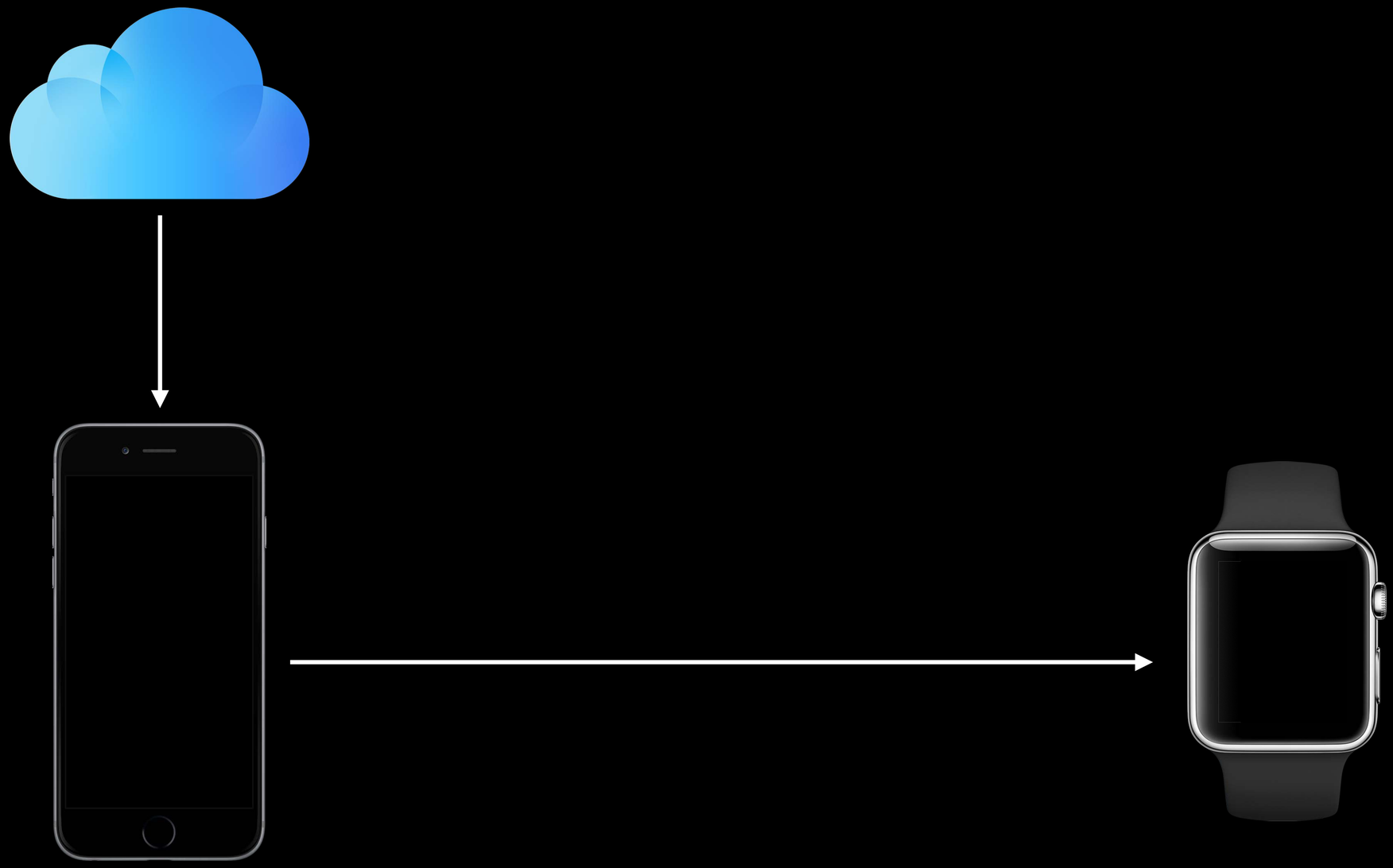
Complications

Pushed: Summary



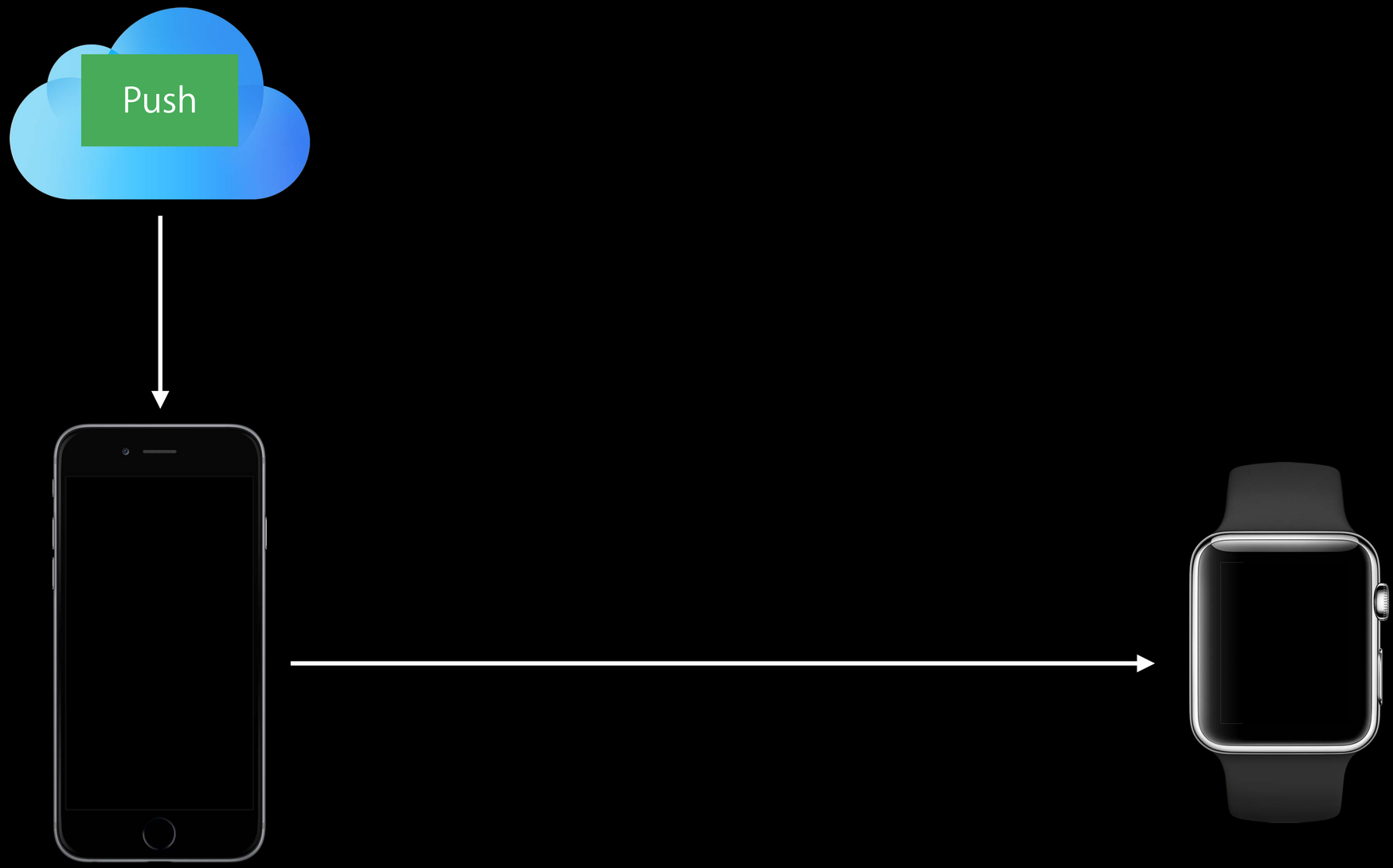
Complications

Pushed: Summary



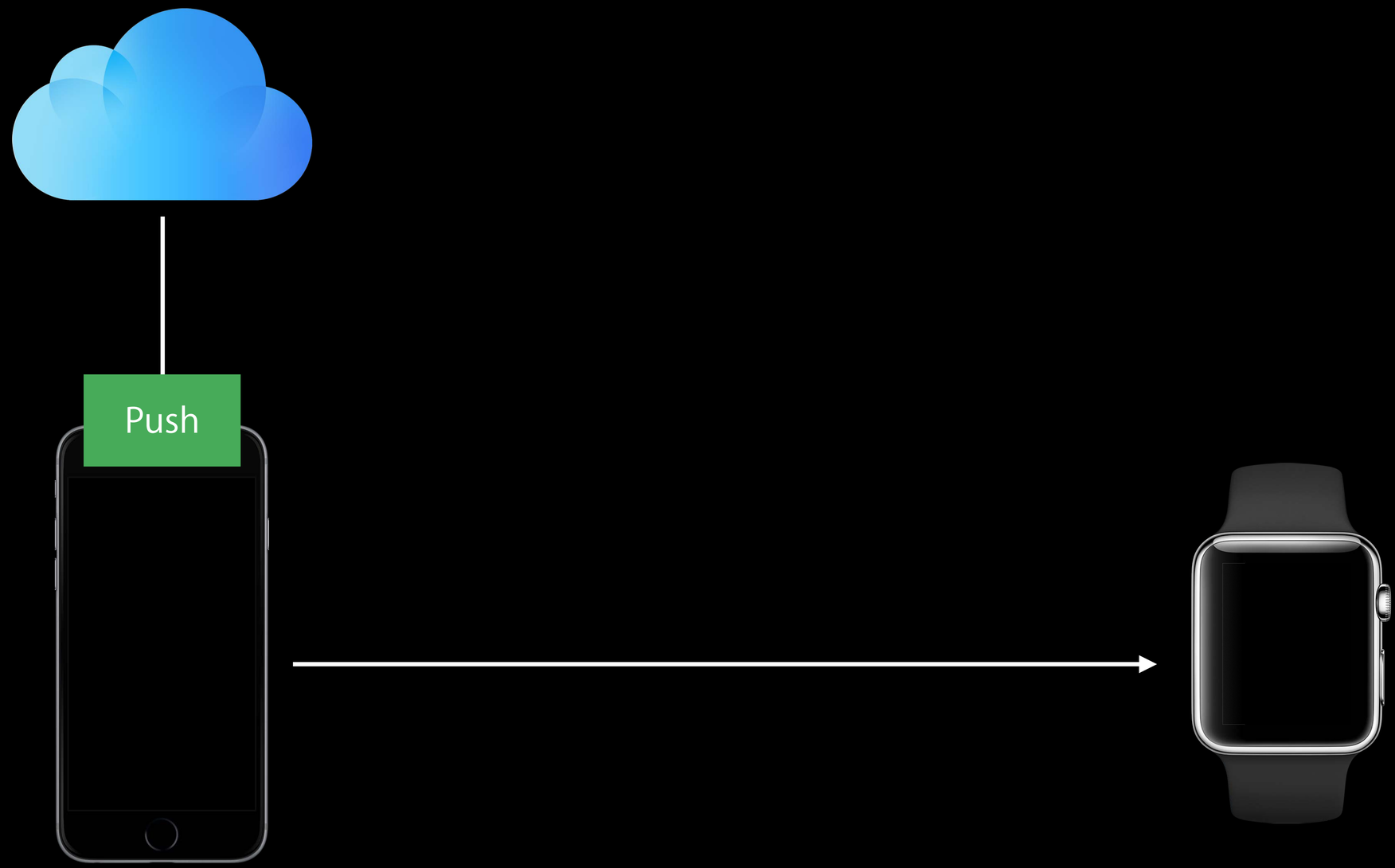
Complications

Pushed: Summary



Complications

Pushed: Summary



Complications

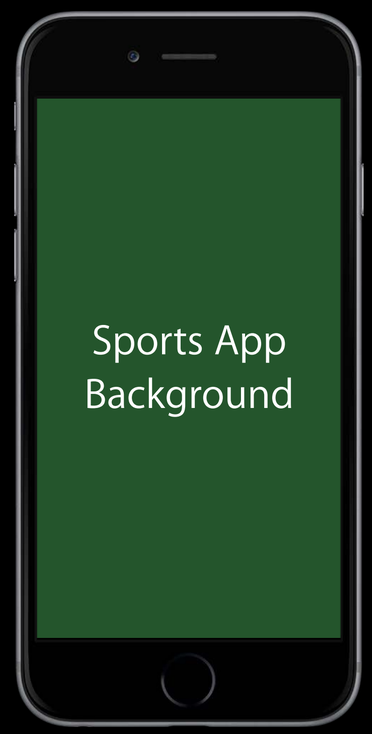
Pushed: Summary

Complications

Pushed: Summary

`transferUserInfo(_:)`

U3

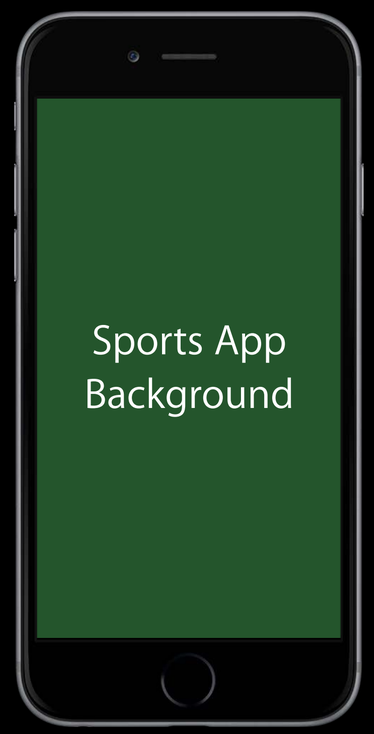


Outstanding User Info Transfers

U2 U1

Complications

Pushed: Summary



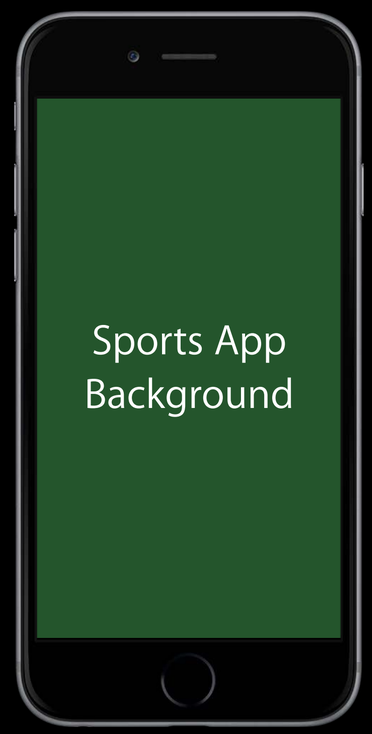
Outstanding User Info Transfers

Complications

Pushed: Summary

```
transferCurrent
ComplicationUserInfo(_:)
```

U4



Outstanding User Info Transfers

U3 U2 U1

Complications

Pushed: Summary

Outstanding User Info Transfers

U3 U2 U1

U4

Complications

Pushed: Summary

Outstanding User Info Transfers

Complications

Pushed: Summary

Outstanding User Info Transfers

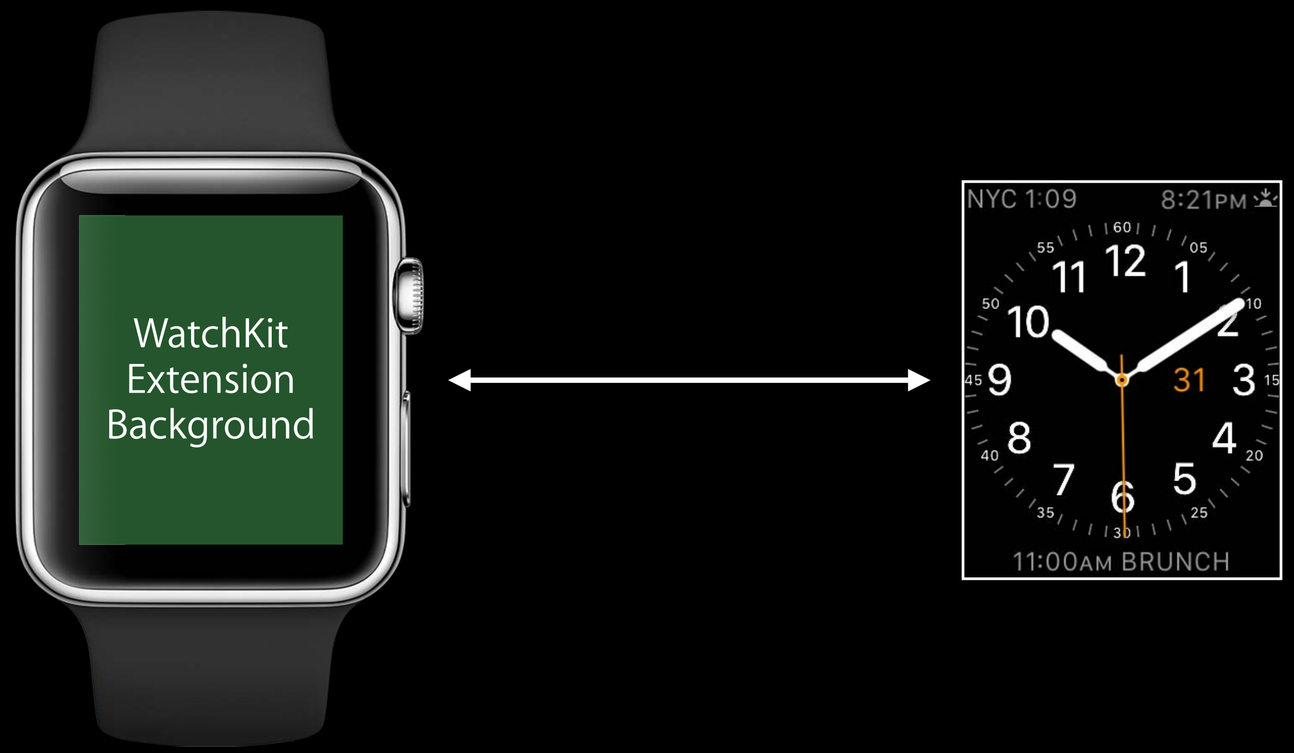
Complications

Pushed: Summary

Outstanding User Info Transfers

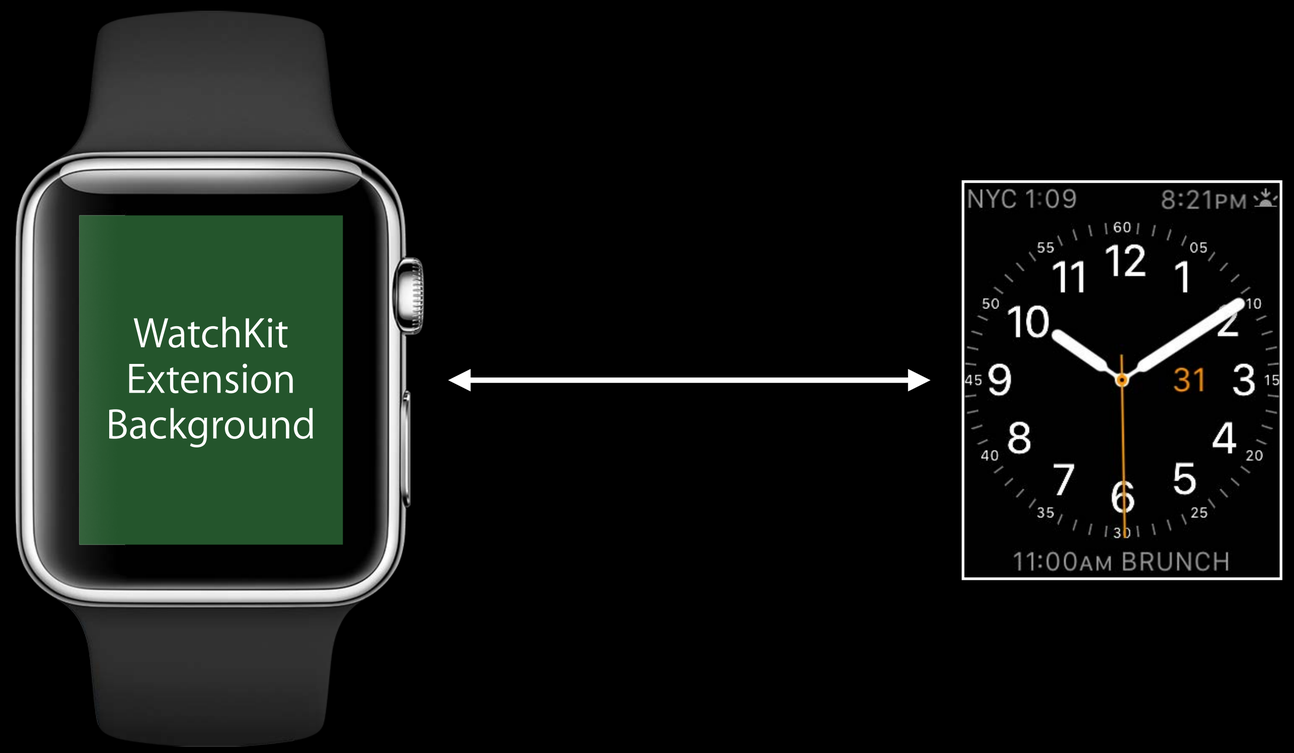
Complications

Pushed: Summary



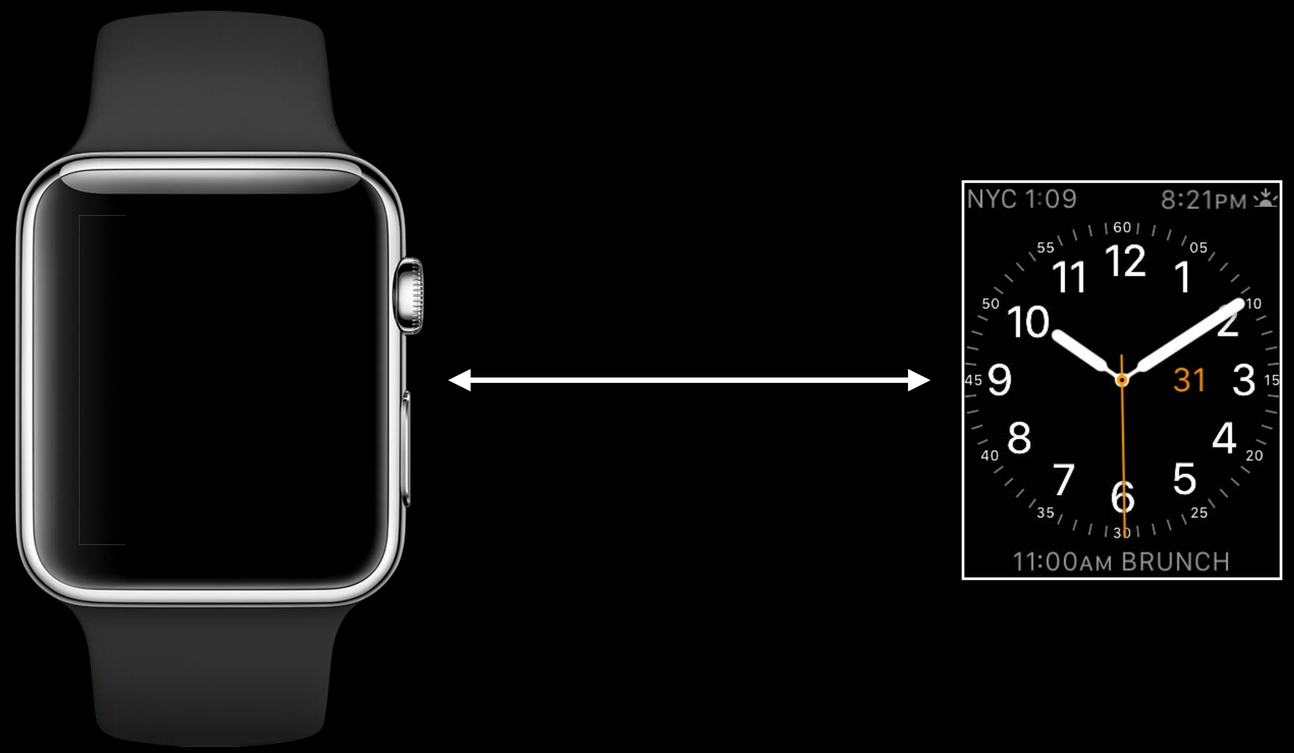
Complications

Pushed: Summary



Complications

Pushed: Summary



Complications

Pushed: Summary

Complications

Pushed: Summary

PushKit

Complications

Pushed: Summary

PushKit

PKPushTypeComplication

Complications

Pushed: Summary

PushKit

PKPushTypeComplication

Complication active on clock face

Complications

Pushed: Summary

PushKit

PKPushTypeComplication

Complication active on clock face

Limited number of pushes per day

Complications

Pushed: Summary

PushKit

PKPushTypeComplication

Complication active on clock face

Limited number of pushes per day

transferUserInfo()

Complications

Pushed: Summary

PushKit

PKPushTypeComplication

Complication active on clock face

Limited number of pushes per day

transferUserInfo()

transferCurrentComplicationUserInfo()

Complications

Pushed: Summary

PushKit

PKPushTypeComplication

Complication active on clock face

Limited number of pushes per day

transferUserInfo()

transferCurrentComplicationUserInfo()

ClockKit

Complications

Pushed: Summary

PushKit

PKPushTypeComplication

Complication active on clock face **BUDGETED**

Limited number of pushes per day **BUDGETED**

transferUserInfo()

transferCurrentComplicationUserInfo() **BUDGETED**

ClockKit

Wrap Up

Summary

WatchConnectivity

NSURLSession

Complications

More Information

Documentation

watchOS 2 Transition Guide

WatchKit Programming Guide

Technical Support

Apple Developer Forums

Developer Technical Support

Sample Code

Lister

WatchKit Catalog

http://developer.apple.com/watchOS

General Inquiries

Jake Behrens,
watchOS Frameworks Evangelist
behrens@apple.com

Related Sessions

Introducing WatchKit for watchOS 2	Presidio	Tuesday 10:00AM
Privacy and Your App	Pacific Heights	Tuesday 2:30PM
Building Watch Apps	Pacific Heights	Tuesday 4:30PM
Security and Your Apps	Mission	Tuesday 4:30PM
Creating Complications with ClockKit	Pacific Heights	Wednesday 11:00AM
Networking with NSURLSession	Pacific Heights	Thursday 9:00AM
WatchKit Tips and Tricks	Presidio	Friday 10:00AM

Labs

Watch Connectivity Lab	Frameworks Lab B	Thursday 1:30PM
WatchKit and ClockKit Complications Lab	Frameworks Lab A	Friday 1:30PM

WWDC15